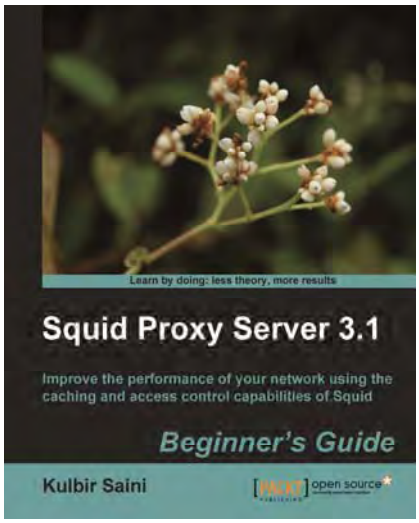
[PACKT] open source
community experience distilled
PUBLISHING

Squid Proxy Server 3.1

Beginner's Guide

Kulbir Saini



Chapter No.3

"Running Squid"

In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.3 "Running Squid"

A synopsis of the book’s content

Information on where to buy this book

About the Author

Kulbir Saini is an entrepreneur based in Hyderabad, India. He has had extensive experience in managing systems and network infrastructure. Apart from his work as a freelance developer, he provides services to a number of startups. Through his blogs, he has been an active contributor of documentation for various open source projects, most notable being The Fedora Project and Squid. Besides computers, which his life practically revolves around, he loves travelling to remote places with his friends. For more details, please check `http://saini.co.in/`.

There are people who served as a source of inspiration, people who helped me throughout, and my friends who were always there for me. Without them, this book wouldn't have been possible.

I would like to thank Sunil Mohan Ranta, Nirnimesh, Suryakant Pati dar, Shiben Bhatt acharjee, Tarun Jain, Sanyam Sharma, Jayaram Kowta,

AmalRaj, Sachin Rawat, Vidit Bansal, Upasana Tegta, Gopal Datt Joshi, Vardhman Jain, Sandeep Chandna, Anurag Singh Rana, Sandeep Kumar, Rishabh Mukherjee, Mahaveer Singh Deora, Sambhav Jain, Ajay Somani, Ankush Kalkote, Deepak Vig, Kapil Agrawal, Sachin Goyal, Pankaj Saini, Alok Kumar,Niti n Bansal, Niti n Gupta, Kapil Bajaj, Gaurav Kharkwal, Atul Dwivedi, Abhinav Parashar, Bhargava Chowdary, Maruti Borker, Abhilash I, Gopal Krishna Koduri, Sashidhar Guntury, Siva Reddy, Prashant Mathur, Vipul Mitt al, Deepti G.P., Shikha Aggarwal, Gaganpreet Singh Arora, Sanrag Sood, Anshuman Singh, Himanshu Singh, Himanshu Sharma, Dinesh Yadav, Tushar Mahajan, Sankalp Khare, Mayank Juneja, Ankur Goel, Anuraj Pandey, Rohit Nigam, Romit Pandey, Ankit Rai, Vishwajeet Singh, Suyesh Tiwari, Sanidhya Kashap, and Kunal Jain.

I would also like to thank Michelle Quadros, Sarah Cullington, Susmita Panda, Priya Mukherji, and Snehman K Kohli from Packt who have been extremely helpful and encouraging during the writing of the book.

Special thanks go out to my parents and sister, for their love and support.

Squid Proxy Server 3.1

Beginner's Guide

Squid proxy server enables you to cache your web content and return it quickly on subsequent requests. System administrators oft en struggle with delays and too much bandwidth being used, but Squid solves these problems by handling requests locally. By deploying Squid in accelerator mode, requests are handled faster than on normal web servers, thus making your site perform quicker than everyone else's!

The Squid Proxy Server 3.1 Beginner's Guide will help you to install and configure Squid so that it is optimized to enhance the performance of your network. Caching usually takes a lot of professional know-how, which can take time and be very confusing. The Squid proxy server reduces the amount of effort that you will have to spend and this book will show you how best to use Squid, saving your time and allowing you to get most out of your network.

Whether you only run one site, or are in charge of a whole network, Squid is an invaluable tool which improves performance immeasurably. Caching and performance optimization usually requires a lot of work on the developer's part, but Squid does all that for you. This book will show you how to get the most out of Squid by customizing it for your network.

You will learn about the different configuration options available and the transparent and accelerated modes that enable you to focus on particular areas of your network. Applying proxy servers to large networks can be a lot of work as you have to decide where to place restrictions and who to grant access. However, the straightforward examples in this book will guide you through step-by-step so that you will have a proxy server that covers all areas of your network by the time you finish reading.

What This Book Covers

Chapter 1, *Getting Started with Squid*, discusses the basics of proxy servers and web caching and how we can utilize them to save bandwidth and improve the end user's browsing experience. We will also learn to identify the correct Squid version for our environment. We will explore various configuration options available for enabling or disabling certain features while we compile Squid from the source code. We will explore steps to compile and install Squid.

Chapter 2, *Configuring Squid*, explores the syntax used in the Squid configuration fi le, which is used to control Squid's behavior. We will explore the important directives used in the configuration fi le and will see related examples to understand them better. We will have a brief overview of the powerful access control lists which we will learn in detail in

later chapters. We will also learn to fi ne-tune our cache to achieve a better HIT ratio to save bandwidth and reduce the average page load time.

Chapter 3, *Running Squid*, talks about running Squid in different modes and various command line options available for debugging purposes. We will also learn about rotating Squid logs to reclaim disk space by deleting old/obsolete log fi les. We will learn to install the `init` script to automatically start Squid on system startup.

Chapter 4, *Getting Started with Squid's Powerful ACLs and Access Rules*, explores the Access Control Lists in detail with examples. We will learn about various ACL types and to construct ACLs to identify requests and responses based on different criteria. We will also learn about mixing ACLs of various types with access rules to achieve desired access control.

Chapter 5, *Understanding Log Files and Log Formats*, discusses configuring Squid to generate customized log messages. We will also learn to interpret the messages logged by Squid in various log fi les.

Chapter 6, *Managing Squid and Monitoring Traffic*, explores the Squid's **Cache Manager** web interface in this chapter using which we can monitor our Squid proxy server and get statistics about different components of Squid. We will also have a look at a few log fi le analyzers which make analyzing traffic simpler compared to manually interpreting the access log messages.

Chapter 7, *Protecting your Squid with Authentication*, teaches us to protect our Squid proxy server with authentication using the various authentication schemes available. We will also learn to write custom authentication helpers using which we can build our own authentication system for Squid.

Chapter 8, *Building a Hierarchy of Squid Caches*, explores cache hierarchies in detail. We will also learn to configure Squid to act as a parent or a sibling proxy server in a hierarchy, and to use other proxy servers as a parent or sibling cache.

Chapter 9, *Squid in Reverse Proxy Mode*, discusses how Squid can accept HTTP requests on behalf of one or more web servers in the background. We will learn to configure Squid in reverse proxy mode. We will also have a look at a few example scenarios.

Chapter 10, *Squid in Intercept Mode*, talks about the details of intercept mode and how to configure the network devices, and the host operating system to intercept the HTTP requests and forward them to Squid proxy server. We will also have a look at the pros and cons of Squid in intercept mode.

Chapter 11, *Writing URL Redirectors and Rewriters*. Squid's behavior can be further customized using the URL redirectors and rewriter helpers. In this chapter, we will learn about the internals of redirectors and rewriters and we will create our own custom helpers.

Chapter 12, *Troubleshooting Squid*, discusses some common problems or errors which you may come across while configuring or running Squid. We will also learn about getting online help to resolve issues with Squid and fi ling bug reports.

3
Running Squid

In the previous chapters, we had learned about compiling, installing, and configuring the Squid proxy server. In this chapter, we are going to learn about the different ways of running Squid and the available options that can be passed to Squid from the command line. We will also learn about debugging the Squid configuration file.

In this chapter, we will learn the following:

- Various command line options for running Squid
- Parsing the squid configuration file for syntax errors
- Using an alternate squid configuration file for testing purposes
- Different ways of starting Squid
- Rotating log files generated by Squid

Let's get started and explore the previous points.

Command line options

Normally, all of the Squid configuration options reside with in the `squid.conf` file (the main Squid configuration file). To tweak the Squid functionality, the preferred method is to change the options in the `squid.conf` file. However there are some options which can also be controlled using additional command line options while running Squid.

These options are not very popular and are rarely used, but these are very useful for debugging problems without the Squid proxy server. Before exploring the command line options, let's see how Squid is run from the command line.

As we saw in the first chapter, the location of the Squid binary file depends on the `--prefix` option passed to the `configure` command before compiling. So, depending upon the value of the `--prefix` option, the location of the Squid executable may be one of `/usr/local/sbin/squid` or `${prefix}/sbin/squid`, where `${prefix}` is the value of the option `--prefix` passed to the `configure` command. Therefore, to run Squid, we need to run one of the following commands on the terminal:

- When the `--prefix` option was not used with the `configure` command, the default location of the Squid executable will be `/usr/local/sbin/squid`.
- When the `--prefix` option was used and was set to a directory, then the location of the Squid executable will be `${prefix}/sbin/squid`.

It's painful to type the absolute path for Squid to run. So, to avoid typing the absolute path, we can include the path to the Squid executable in our PATH shell variable, using the `export` command as shown in the following example:

```
$ export PATH=$PATH:/usr/local/sbin/
```

Alternatively, we can use the following command:

```
$ export PATH=$PATH:/opt/squid/sbin/
```

We can also add the preceding command to our `~/.bashrc` or `~/.bash_profile` file to avoid running the `export` command every time we enter a new shell.

After setting the PATH shell variable appropriately, we can run Squid by simply typing the following command on shell:

```
$ squid
```

This command will run Squid after loading the configuration options from the `squid.conf` file.

We'll be using the `squid` command without an absolute path for running the Squid process. Please use the appropriate path according to the installation prefix which you have chosen.

Now that we know how to run Squid from the command line, let's have a look at the various command line options.

Getting a list of available options

Before actually moving forward, we should firstly check the available set of options for our Squid installation.

Time for action – listing the options

Like a lot of other Linux programs, Squid also provides the option `-h` which can be used to retrieve a list of options:

```
squid -h
```

The previous command will result in the following output:

```
Usage: squid [-cdhvzCFNRVYX] [-s | -l facility] [-f config-file] [-[au]
port] [-k signal]
   -a port   Specify HTTP port number (default: 3128).
   -d level  Write debugging to stderr also.
   -f file   Use given config-file instead of
             /opt/squid/etc/squid.conf.
   -h  Print help message.
   -k  reconfigure|rotate|shutdown|interrupt|kill|debug|check|parse
          Parse configuration file, then send signal to
          running copy (except -k parse) and exit.
   -s | -l facility
          Enable logging to syslog.
   -u  port Specify ICP port number (default: 3130), disable with 0.
   -v  Print version.
   -z  Create swap directories.
   -C  Do not catch fatal signals.
   -F  Don't serve any requests until store is rebuilt.
   -N  No daemon mode.
   -R  Do not set REUSEADDR on port.
   -S  Double-check swap during rebuild.
...
```

We will now have a look at a few important options from the preceding list. We will also, have a look at the `squid(8)` man page or `http://linux.die.net/man/8/squid` for more details.

What just happened?

We have just used the `squid` command to list the available options which we can use on the command line.

Getting information about our Squid installation

Various features may vary across different versions of Squid. Before proceeding any further, it's a good idea to know the version of Squid installed on our machine.

Time for action – finding out the Squid version

Just in case we want to check which version of Squid we are using or the options we used with the `configure` command before compiling, we can use the option `-v` on the command line. Let's run Squid with this option:

```
squid -v
```

If we try to run the preceding command in the terminal, it will produce an output similar to the following:

```
Squid Cache: Version 3.1.10
configure options:  '--config-cache' '--prefix=/opt/squid/' '--enable-
storeio=ufs,aufs' '--enable-removal-policies=lru,heap' '--enable-icmp'
'--enable-useragent-log' '--enable-referer-log' '--enable-cache-digests'
'--with-large-files' --enable-ltdl-convenience
```

What just happened?

We used the `squid` command with the `-v` option to find out the version of Squid installed on our machine, and the options used with the `configure` command before compiling Squid.

Creating cache or swap directories

As we learned in the previous chapter, the cache directories specified using the `cache_dir` directive in the `squid.conf` file, must already exist before Squid can actually use them.

Time for action – creating cache directories

Squid provides the `-z` command line option to create the swap directories. Let's see an example:

```
squid -z
```

If this option is used and the cache directories don't exist already, the output will look similar to the following:

```
2010/07/20 21:48:35| Creating Swap Directories
2010/07/20 21:48:35| Making directories in /squid_cache/00
```

```
2010/07/20 21:48:35| Making directories in /squid_cache/01
2010/07/20 21:48:35| Making directories in /squid_cache/02
2010/07/20 21:48:35| Making directories in /squid_cache/03
...
```

We should use this option whenever we add new cache directories in the Squid configuration file.

What just happened?

When the `squid` command is run with the option `-z`, Squid reads all the cache directories from the configuration file and checks if they already exist. It will then create the directory structure for all the cache directories that don't exist.

Have a go hero – adding cache directories

Add two or three test cache directories with different values of level 1 (8, 16, and 32) and level 2 (64, 256, and 512) to the configuration file. Then try creating them using the `squid` command. Now study the difference in the directory structure.

Using a different configuration file

The default location for Squid's main configuration file is `${prefix}/etc/squid/squid.conf`. Whenever we run Squid, the main configuration is read from the default location. While testing or deploying a new configuration, we may want to use a different configuration file just to check whether it will work or not. We can achieve this by using the option `-f`, which allows us to specify a custom location for the configuration file. Let's see an example:

```
squid -f /etc/squid.minimal.conf
# OR
squid -f /etc/squid.alternate.conf
```

If Squid is run this way, Squid will try to load the configuration from `/etc/squid.minimal.conf` or `/etc/squid.alternate.conf`, and it will completely ignore the `squid.conf` from the default location.

Getting verbose output

When we run Squid from the terminal without any additional command line options, only warnings and errors are displayed on the terminal (or `stderr`). However, while testing, we would like to get a verbose output on the terminal, to see what is happening when Squid starts up.

Time for action – debugging output in the console

To get more information on the terminal, we can use the option `-d`. The following is an example:

```
squid -d 2
```

We must specify an integer with the option `-d` to indicate the verbosity level. Let's have a look at the meaning of the different levels:

- Only critical and fatal errors are logged when level `0` (zero) is used.
- Level `1` includes the logging of important problems.
- Level `2` and higher includes the logging of informative details and other actions.

Higher levels result in more output on the terminal. A sample output on the terminal with level `2` would look similar to the following:

```
2010/07/20 21:40:53| Starting Squid Cache version 3.1.10 for i686-pc-
linux-gnu...
2010/07/20 21:40:53| Process ID 15861
2010/07/20 21:40:53| With 1024 file descriptors available
2010/07/20 21:40:53| Initializing IP Cache...
2010/07/20 21:40:53| DNS Socket created at [::], FD 7
2010/07/20 21:40:53| Adding nameserver 192.168.36.222 from /etc/resolv.
conf
2010/07/20 21:40:53| User-Agent logging is disabled.
2010/07/20 21:40:53| Referer logging is disabled.
2010/07/20 21:40:53| Unlinkd pipe opened on FD 13
2010/07/20 21:40:53| Local cache digest enabled; rebuild/rewrite every
3600/3600 sec
2010/07/20 21:40:53| Store logging disabled
2010/07/20 21:40:53| Swap maxSize 0 + 262144 KB, estimated 20164 objects
2010/07/20 21:40:53| Target number of buckets: 1008
2010/07/20 21:40:53| Using 8192 Store buckets
2010/07/20 21:40:53| Max Mem  size: 262144 KB
2010/07/20 21:40:53| Max Swap size: 0 KB
2010/07/20 21:40:53| Using Least Load store dir selection
2010/07/20 21:40:53| Current Directory is /opt/squid/sbin
2010/07/20 21:40:53| Loaded Icons.
```

```
2010/07/20 21:40:53| Accepting  HTTP connections at [::]:3128, FD 14.
2010/07/20 21:40:53| HTCP Disabled.
2010/07/20 21:40:53| Squid modules loaded: 0
2010/07/20 21:40:53| Ready to serve requests.
2010/07/20 21:40:54| storeLateRelease: released 0 objects
...
```

As we can see, Squid is trying to dump a log of actions that it is performing. The messages shown are mostly startup messages and there will be fewer messages when Squid starts accepting connections.

Starting Squid in debug mode is quite helpful when Squid is up and running and users complain about poor speeds or being unable to connect. We can have a look at the debugging output and the appropriate actions to take.

What just happened?

We started Squid in debugging mode and can now see Squid writing an output on the command line, which is basically a log of the actions which Squid is performing. If Squid is not working, we'll be able to see the reasons on the command line and we'll be able to take actions accordingly.

Full debugging output on the terminal

The option `-d` specifies the verbosity level of the output dumped by Squid on the terminal. If we require all of the debugging information on the terminal, we can use the option `-X`, which will force Squid to write debugging information at every single step. If the option `-X` is used, we'll see information about parsing the `squid.conf` file and the actions taken by Squid, based on the configuration directives encountered. Let's see a sample output produced when option `-X` is used:

```
...
2010/07/21 21:50:51.515| Processing: 'acl my_machines src 172.17.8.175
10.2.44.46 127.0.0.1 172.17.11.68 192.168.1.3'
2010/07/21 21:50:51.515| ACL::Prototype::Registered: invoked for type src
2010/07/21 21:50:51.515| ACL::Prototype::Registered:    yes
2010/07/21 21:50:51.515| ACL::FindByName 'my_machines'
2010/07/21 21:50:51.515| ACL::FindByName found no match
2010/07/21 21:50:51.515| aclParseAclLine: Creating ACL 'my_machines'
2010/07/21 21:50:51.515| ACL::Prototype::Factory: cloning an object for
type 'src'
```

```
2010/07/21 21:50:51.515| aclParseIpData: 172.17.8.175
2010/07/21 21:50:51.515| aclParseIpData: 10.2.44.46
2010/07/21 21:50:51.515| aclParseIpData: 127.0.0.1
2010/07/21 21:50:51.515| aclParseIpData: 172.17.11.68
2010/07/21 21:50:51.515| aclParseIpData: 192.168.1.3
...
```

Let's see what this output means. In the first line, Squid encountered a line defining an ACL `my_machines`. The next few lines in the output describe Squid invoking different methods to parse, creating a new ACL, and then assigning values to it. This option can be very helpful while debugging ambiguous ACLs.

Running as a normal process

Sometime during testing, we may not want Squid to run as a daemon. Instead, we may want it to run as a normal process which we can interrupt easily by pressing `CTRL-C`. To achieve this, we can use the option `-N`. When this option is used, Squid will not run in the background it will run in the current shell instead.

Parsing the Squid configuration file for errors or warnings

It's a good idea to parse or check the configuration file (`squid.conf`) for any errors or warnings before we actually try to run Squid, or reload a Squid process which is already running in a production deployment. Squid provides an option `-k` with an argument `parse`, which, if supplied, will force Squid to parse the current Squid configuration file and report any errors and warnings. `Squid -k` is also used to check and report directive and option changes when we upgrade our Squid version.

Time for action – testing our configuration file

As we learned before, we can use the `-k parse` option to test our configuration file. Now, we are going to add a test line and see if Squid can catch the error.

1. For example, let's add the following line to our `squid.conf` file:
   `unknown_directive 1234`
2. Now we'll run Squid with the `-k parse` option as follows:
   `squid -k parse`

3. As `unknown_directive` is not a valid directive for the Squid configuration file, we should get an error similar to the following:

```
2010/07/21 22:28:40| cache_cf.cc(346) squid.conf:945 unrecognized:
'unknown_directive'
```

So, if we find an error within our configuration file, we can go back and fix the errors and then parse the configuration file again.

What just happened?

We first added an invalid line in to our configuration file and then tried to parse it using a `squid` command which resulted in an error. It is a good idea to always parse the configuration file before starting Squid.

Sending various signals to a running Squid process

Squid provides the `-k` option to send various signals to a Squid process which is already running. Using this option, we can send various management signals such as, reload the configuration file, rotate the log files, shut down the proxy server, switch to debug mode, and many more. Let's have a look at some of the important signals which are available.

Please note that when the argument `parse` is used with the option `-k`, no signal is sent to the running Squid process.

Reloading a new configuration file in a running process

We may need to make changes to our Squid configuration file sometimes, even when it is deployed in production mode. In such cases, after making changes, we don't want to restart our proxy server because that will introduce a significant downtime and will also interrupt active connections between clients and remote servers. In these situations, we can use the option `-k` with `reconfigure` as an argument, to signal Squid to re-read the configuration file and apply the new settings:

```
squid -k reconfigure
```

The previous command will force Squid to re-read the configuration file, while serving the requests normally and not terminating any active connections.

It's good practice to parse the configuration file for any errors or warning using the `-k parse` option before issuing the `reconfigure` signal.

Shutting down the Squid process

To shut down a Squid process which is already running, we can issue a `shutdown` signal with the help of the option `-k` as follows:

```
squid -k shutdown
```

Squid tries to terminate connections gracefully when it receives a `shutdown` signal. It will allow the active connection to finish before the process is completely shut down or terminated.

Interrupting or killing a running Squid process

If we have a lot of clients, Squid can take a significant amount of time before it completely terminates itself on receiving the `-k shutdown` signal. To get Squid to immediately stop serving requests, we can use the `-k interrupt` signal. The `-k interrupt` signal will not allow Squid to wait for active connections to finish and will stop the process immediately.

In some cases, the Squid process may not be stopped using `-k shutdown` or `-k interrupt` signals. If we want to terminate the process immediately, we can issue a `-k kill` signal, which will immediately kill the process. This signal should only be used when Squid can't be stopped with `-k shutdown` or `-k interrupt`. For example, to send a `-k kill` signal to Squid, we can use the following command:

```
squid -k kill
```

This command will kill the Squid process immediately.

Please note that `shutdown`, `interrupt`, and `kill` are Squid signals and not the system kill signals which are emulated.

Checking the status of a running Squid process

To know whether a Squid process is already running or not, we can issue a `check` signal which will tell us the status of the Squid process. Squid will also validate the configuration file and report any fatal problems. If the Squid process is running fine, the following command will exit without printing anything:

```
squid -k check
```

Otherwise, if the process has exited, this command will give an error similar to the following error message:

```
squid: ERROR: Could not send signal 0 to process 25243: (3) No such
process
```

Have a go hero – check the return value

After running `squid -k check`, find out the return value or status in scenarios when:

- Squid was running
- Squid was not running

Solution: The return value or the status of a command can be found out by using the command `echo $?`. This will print the return status or value of the previous command that was executed in the same shell. Return values should be (1) -> 0, (2) -> 1.

Sending a running process in to debug mode

If we didn't start Squid in debug mode and for testing we don't want to stop an already running Squid process, we can issue a debug signal which will send the already running process into debug mode. The debugging output will then be written to Squid's `cache.log` file located at `${prefix}/var/logs/cache.log` or `/var/log/squid/cache.log`.

The Squid process running in debug mode may write a log of debugging output to the `cache.log` file and may quickly consume a lot of disk space.

Rotating the log files

The log files used by Squid grow in size over a period of time and can consume a significant amount of disk space. To get rid of the older logs, Squid provides a `rotate` signal that can be issued when we want to rotate the existing log files. Upon receiving this signal, Squid will close the current log files, move them to other filenames, or delete them based on the configuration directive `logfile_rotate` (in `squid.conf`) then reopen the files to write the logs.

It's quite inconvenient to rotate log files manually. So, we can automate the process of log file rotation with the help of a cron job. Let's say we want to rotate the log files at midnight, the corresponding cron tab entry will be:

```
59 23 * * * /opt/squid/sbin/squid -k rotate
```

Please note that the path to Squid executable may differ depending on the installation prefix. We'll learn more about log files in *Chapter 5*, *Understanding Log Files and Log Formats*.

Forcing the storage metadata to rebuild

When Squid starts, it tries to load the storage metadata. If Squid fails to load the storage metadata, then it will try to rebuild it. If it receives any requests during that period, Squid will try to satisfy those requests in parallel, which results in slow rebuild. We can force Squid to rebuild the metadata before it starts processing any requests using the option `-F` on the command line. This may result in a quick rebuild of storage metadata but clients may have to wait for a significant time, if the cache is large. For large caches, we should try to avoid this option:

```
squid -F
```

Squid will now rebuild the cache metadata and will not serve any client requests until the metadata rebuild process is complete.

Double checking swap during rebuild

The option `-F` determines whether Squid should serve requests while the storage metadata is being rebuilt. We have another option, `-S`, which can be used to force Squid to double check the cache during rebuild. If we use the `-S` option along with the option `-d` as follows:

```
squid -S -d 1
```

This will produce a debugging output on the terminal which will look similar to the following:

```
2010/07/21 21:29:22| Beginning Validation Procedure
2010/07/21 21:29:22| UFSSwapDir::doubleCheck: SIZE MISMATCH
2010/07/21 21:29:22| UFSSwapDir::doubleCheck: ENTRY SIZE: 1332, FILE
SIZE: 114
2010/07/21 21:29:22| UFSSwapDir::dumpEntry: FILENO 00000092
2010/07/21 21:29:22| UFSSwapDir::dumpEntry: PATH /squid_
cache/00/00/00000092
2010/07/21 21:29:22| StoreEntry->key: 0060E9E547F3A1AAEEDE369C5573F8D9
2010/07/21 21:29:22| StoreEntry->next: 0
2010/07/21 21:29:22| StoreEntry->mem_obj: 0
2010/07/21 21:29:22| StoreEntry->timestamp: 1248375049
2010/07/21 21:29:22| StoreEntry->lastref: 1248375754
2010/07/21 21:29:22| StoreEntry->expires: 1279911049
2010/07/21 21:29:22| StoreEntry->lastmod: 1205097338
2010/07/21 21:29:22| StoreEntry->swap_file_sz: 1332
2010/07/21 21:29:22| StoreEntry->refcount: 1
2010/07/21 21:29:22| StoreEntry->flags: CACHABLE,DISPATCHED
2010/07/21 21:29:22| StoreEntry->swap_dirn: 0
```

```
2010/07/21 21:29:22| StoreEntry->swap_filen: 146
2010/07/21 21:29:22| StoreEntry->lock_count: 0
2010/07/21 21:29:22| StoreEntry->mem_status: 0
...
```

Squid is basically trying to validate each and every cached object on the disk.

Automatically starting Squid at system startup

Once we have a properly configured and running proxy server, we would like it to start whenever the system is started or rebooted. Next, we'll have a brief look at the most common ways of adding or modifying the boot scripts for popular operating systems. These methods will most probably work on your operating system. If they don't, please refer to the corresponding operating system manual for information on boot scripts.

Adding Squid command to /etc/rc.local file

Adding the full path of the Squid executable file is the easiest way to start Squid on system startup. The file `/etc/rc.local` is executed after the system boots as the super or root user. We can place the Squid command in this file and it will run every time the system boots up. Add the following line at the end of the `/etc/rc.local` file:

```
${prefix}/sbin/squid
```

Please replace `${prefix}` with the installation prefix which you used before compiling Squid.

Adding init script

Alternatively, we can add a simple init script which will be a simple shell script to start the Squid process or send various signals to a running Squid process. Init scripts are supported by most operating systems and are generally located at `/etc/init.d/`, `/etc/rc.d/`, or `/etc/rc.d/init.d/`. Any shell script placed in any of these directories is executed at system startup with root privileges.

Time for action – adding the init script

We are going to use a simple shell script, as shown in the following example, which takes a single command line argument and acts accordingly:

```
#!/bin/bash
# init script to control Squid server
case "$1" in
```

```
start)
  /opt/squid/sbin/squid
  ;;

stop)
  /opt/squid/sbin/squid -k shutdown
  ;;

reload)
  /opt/squid/sbin/squid -k reconfigure
  ;;

restart)
  /opt/squid/sbin/squid -k shutdown
  sleep 2
  /opt/squid/sbin/squid
  ;;

*)
  echo $"Usage: $0 {start|stop|reload|restart}"
  exit 2
esac
exit $?
```

Please note the absolute path to the Squid executable here and change it accordingly. We can save this shell script to a file with the name `squid` and then move it to one of the directories we discussed earlier depending on our operating system.

The Squid source carries an `init` script located at `contrib/squid.rc`, but it's installed only on a few systems by default.

What just happened?

We added an `init` script to control the Squid proxy server. Using this script, we can start, stop, restart, or reload the process. It's important that the script is placed in the correct directory, otherwise the Squid proxy server will not start on system startup.

Pop quiz

1. What should be the first step undertaken after adding new cache directories to the configuration file?
   a. Reboot the server.
   b. Run the `squid` command with the `-z` option.
   c. Do nothing, Squid will take care of everything by itself.
   d. Run Squid with root privileges.
2. Where should we look for errors in case Squid is not running properly?
   a. Access log file.
   b. Cache log file.
   c. Squid configuration file.
   d. None of the above.
3. In which scenario should we avoid debug mode?
   a. While testing the server.
   b. While Squid is deployed in production mode.
   c. When we have a lot of spare disk space.
   d. When we have a lot of spare RAM.

Summary

In this chapter, we learned about the various command line options which can be used while running Squid, how to start the Squid process in a different mode, and how to send signals to a process which is already running. We also learned about creating new cache directories after adding them to the Squid configuration file.

We specifically covered the following:

- Parsing the Squid configuration file for errors and warnings.
- Using various options to get suitable debugging outputs while testing.
- Reloading a new configuration in a Squid process which is already running, without interrupting service.
- Automatic rotation of log files to recover disk space.

We also learned about configuring our system to start a Squid process whenever the system boots up.

Now that we have learned about running a Squid process, we're ready to explore access control lists in detail and test them on a live Squid server.

Where to buy this book

You can buy Squid Proxy Server 3.1 Beginner's Guide from the Packt Publishing website: `https://www.packtpub.com/squid-proxy-server-3-1-beginners-guide/book`

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

www.PacktPub.com

For More Information:
www.packtpub.com/squid-proxy-server-3-1-beginners-guide/book