Stopping and Restarting Apache HTTP Server

This document covers stopping and restarting Apache HTTP Server on Unix-like systems. Windows NT, 2000 and XP users should see Running httpd as a Service and Windows 9x and ME users should see Running httpd as a Console Application for information on how to control httpd on those platforms.

See also

- httpd
- apachectl
- Starting
- Comments

Introduction

In order to stop or restart the Apache HTTP Server, you must send a signal to the running httpd processes. There are two ways to send the signals. First, you can use the unix `kill` command to directly send signals to the processes. You will notice many httpd executables running on your system, but you should not send signals to any of them except the parent, whose pid is in the PidFile. That is to say you shouldn't ever need to send signals to any process except the parent. There are four signals that you can send the parent: TERM, USR1, HUP, and WINCH, which will be described in a moment.

To send a signal to the parent you should issue a command such as:

```
kill -TERM `cat /usr/local/apache2/logs/httpd.pid`
```

The second method of signaling the httpd processes is to use the `-k` command line options: `stop`, `restart`, `graceful` and `graceful-stop`, as described below. These are arguments to the httpd binary, but we recommend that you send them using the apachectl control script, which will pass them through to httpd.

After you have signaled httpd, you can read about its progress by issuing:

```
tail -f /usr/local/apache2/logs/error_log
```

Modify those examples to match your ServerRoot and PidFile settings.

Stop Now

Signal: TERM

```
apachectl -k stop
```

Sending the `TERM` or `stop` signal to the parent causes it to immediately attempt to kill off all of its children. It may take it several seconds to complete killing off its children. Then the parent itself exits. Any requests in progress are terminated, and no further requests are served.

Graceful Restart

Signal: USR1

```
apachectl -k graceful
```

The `USR1` or `graceful` signal causes the parent process to *advise* the children to exit after their current request (or to exit immediately if they're not serving anything). The parent re-reads its configuration files and re-opens its log files. As each child dies off the parent replaces it with a child from the new *generation* of the configuration, which begins serving new requests immediately.

This code is designed to always respect the process control directive of the MPMs, so the number of processes and threads available to serve clients will be maintained at the appropriate values throughout the restart process. Furthermore, it respects StartServers in the following manner: if after one second at least

StartServers new children have not been created, then create enough to pick up the slack. Hence the code tries to maintain both the number of children appropriate for the current load on the server, and respect your wishes with the StartServers parameter.

Users of mod_status will notice that the server statistics are **not** set to zero when a USR1 is sent. The code was written to both minimize the time in which the server is unable to serve new requests (they will be queued up by the operating system, so they're not lost in any event) and to respect your tuning parameters. In order to do this it has to keep the *scoreboard* used to keep track of all children across generations.

The status module will also use a G to indicate those children which are still serving requests started before the graceful restart was given.

At present there is no way for a log rotation script using USR1 to know for certain that all children writing the pre-restart log have finished. We suggest that you use a suitable delay after sending the USR1 signal before you do anything with the old log. For example if most of your hits take less than 10 minutes to complete for users on low bandwidth links then you could wait 15 minutes before doing anything with the old log.

> When you issue a restart, a syntax check is first run, to ensure that there are no errors in the configuration files. If your configuration file has errors in it, you will get an error message about that syntax error, and the server will refuse to restart. This avoids the situation where the server halts and then cannot restart, leaving you with a non-functioning server.
>
> This still will not guarantee that the server will restart correctly. To check the semantics of the configuration files as well as the syntax, you can try starting httpd as a non-root user. If there are no errors it will attempt to open its sockets and logs and fail because it's not root (or because the currently running httpd already has those ports bound). If it fails for any other reason then it's probably a config file error and the error should be fixed before issuing the graceful restart.

Restart Now

Signal: HUP

```
apachectl -k restart
```

Sending the HUP or restart signal to the parent causes it to kill off its children like in TERM, but the parent doesn't exit. It re-reads its configuration files, and re-opens any log files. Then it spawns a new set of children and continues serving hits.

Users of mod_status will notice that the server statistics are set to zero when a HUP is sent.

> As with a graceful restart, a syntax check is run before the restart is attempted. If your configuration file has errors in it, the restart will not be attempted, and you will receive notification of the syntax error(s).

Graceful Stop

Signal: WINCH

```
apachectl -k graceful-stop
```

The WINCH or graceful-stop signal causes the parent process to *advise* the children to exit after their current request (or to exit immediately if they're not serving anything). The parent will then remove its PidFile and cease listening on all ports. The parent will continue to run, and monitor children which are handling requests. Once all children have finalised and exited or the timeout specified by the `GracefulShutdownTimeout` has been reached, the parent will also exit. If the timeout is reached, any remaining children will be sent the `TERM` signal to force them to exit.

A `TERM` signal will immediately terminate the parent process and all children when in the "graceful" state. However as the `PidFile` will have been removed, you will not be able to use `apachectl` or `httpd` to send this signal.

> The `graceful-stop` signal allows you to run multiple identically configured instances of `httpd` at the same time. This is a powerful feature when performing graceful upgrades of httpd, however it can also cause deadlocks and race conditions with some configurations.
>
> Care has been taken to ensure that on-disk files such as lock files (`Mutex`) and Unix socket files (`ScriptSock`) contain the server PID, and should coexist without problem. However, if a configuration directive, third-party module or persistent CGI utilises any other on-disk lock or state files, care should be taken to ensure that multiple running instances of `httpd` do not clobber each other's files.
>
> You should also be wary of other potential race conditions, such as using `rotatelogs` style piped logging. Multiple running instances of `rotatelogs` attempting to rotate the same logfiles at the same time may destroy each other's logfiles.