NAME

pg_ctl - initialize, start, stop, or control a PostgreSQL server

SYNOPSIS

pg_ctl init[db] [**-D** *datadir*] [**-s**] [**-o** *initdb-options*]

pg_ctl start [**-D** *datadir*] [**-l** *filename*] [**-W**] [**-t** *seconds*] [**-s**] [**-o** *options*] [**-p** *path*] [**-c**]

pg_ctl stop [**-D** *datadir*] [**-m s[mart]** | **f[ast]** | **i[mmediate]**] [**-W**] [**-t** *seconds*] [**-s**]

pg_ctl restart [**-D** *datadir*] [**-m s[mart]** | **f[ast]** | **i[mmediate]**] [**-W**] [**-t** *seconds*] [**-s**] [**-o** *options*] [**-c**]

pg_ctl reload [**-D** *datadir*] [**-s**]

pg_ctl status [**-D** *datadir*]

pg_ctl promote [**-D** *datadir*] [**-W**] [**-t** *seconds*] [**-s**]

pg_ctl logrotate [**-D** *datadir*] [**-s**]

pg_ctl kill *signal_name process_id*

On Microsoft Windows, also:

pg_ctl register [**-D** *datadir*] [**-N** *servicename*] [**-U** *username*] [**-P** *password*] [**-S a[uto]** | **d[emand]**] [**-e** *source*] [**-W**] [**-t** *seconds*] [**-s**] [**-o** *options*]

pg_ctl unregister [**-N** *servicename*]

DESCRIPTION

pg_ctl is a utility for initializing a PostgreSQL database cluster, starting, stopping, or restarting the PostgreSQL database server (**postgres**(1)), or displaying the status of a running server. Although the server can be started manually, pg_ctl encapsulates tasks such as redirecting log output and properly detaching from the terminal and process group. It also provides convenient options for controlled shutdown.

The **init** or **initdb** mode creates a new PostgreSQL database cluster, that is, a collection of databases that will be managed by a single server instance. This mode invokes the **initdb** command. See **initdb**(1) for details.

start mode launches a new server. The server is started in the background, and its standard input is attached to /dev/null (or nul on Windows). On Unix-like systems, by default, the server's standard output and standard error are sent to pg_ctl's standard output (not standard error). The standard output of pg_ctl should then be redirected to a file or piped to another process such as a log rotating program like rotatelogs; otherwise **postgres** will write its output to the controlling terminal (from the background) and will not leave the shell's process group. On Windows, by default the server's standard output and standard error are sent to the terminal. These default behaviors can be changed by using **-l** to append the server's output to a log file. Use of either **-l** or output redirection is recommended.

stop mode shuts down the server that is running in the specified data directory. Three different shutdown methods can be selected with the **-m** option. "Smart" mode disallows new connections, then waits for all existing clients to disconnect. If the server is in hot standby, recovery and streaming replication will be terminated once all clients have disconnected. "Fast" mode (the default) does not wait for clients to disconnect. All active transactions are rolled back and clients are forcibly disconnected, then the server is shut down. "Immediate" mode will abort all server processes immediately, without a clean shutdown. This choice will lead to a crash-recovery cycle during the next server start.

restart mode effectively executes a stop followed by a start. This allows changing the **postgres** command-line options, or changing configuration-file options that cannot be changed without restarting the server. If relative paths were used on the command line during server start, **restart** might fail unless pg_ctl is executed in the same current directory as it was during server start.

reload mode simply sends the **postgres** server process a SIGHUP signal, causing it to reread its configuration files (postgresql.conf, pg_hba.conf, etc.). This allows changing configuration-file options that do not require a full server restart to take effect.

status mode checks whether a server is running in the specified data directory. If it is, the server's PID and the command line options that were used to invoke it are displayed. If the server is not running, pg_ctl returns an exit status of 3. If an accessible data directory is not specified, pg_ctl returns an exit status of 4.

promote mode commands the standby server that is running in the specified data directory to end standby mode and begin read-write operations.

logrotate mode rotates the server log file. For details on how to use this mode with external log rotation tools, see Section 25.3.

kill mode sends a signal to a specified process. This is primarily valuable on Microsoft Windows which does not have a built-in kill command. Use --help to see a list of supported signal names.

register mode registers the PostgreSQL server as a system service on Microsoft Windows. The **-S** option allows selection of service start type, either "auto" (start service automatically on system startup) or "demand" (start service on demand).

unregister mode unregisters a system service on Microsoft Windows. This undoes the effects of the **register** command.

OPTIONS

-c
--core-files

Attempt to allow server crashes to produce core files, on platforms where this is possible, by lifting any soft resource limit placed on core files. This is useful in debugging or diagnosing problems by allowing a stack trace to be obtained from a failed server process.

-D *datadir*
--pgdata=*datadir*

Specifies the file system location of the database configuration files. If this option is omitted, the environment variable **PGDATA** is used.

-l *filename*
--log=*filename*

Append the server log output to *filename*. If the file does not exist, it is created. The umask is set to 077, so access to the log file is disallowed to other users by default.

-m *mode*
--mode=*mode*

Specifies the shutdown mode. *mode* can be smart, fast, or immediate, or the first letter of one of these three. If this option is omitted, fast is the default.

-o *options*
--options=*options*

Specifies options to be passed directly to the **postgres** command. **-o** can be specified multiple times, with all the given options being passed through.

The *options* should usually be surrounded by single or double quotes to ensure that they are passed through as a group.

-o *initdb-options*
--options=*initdb-options*

Specifies options to be passed directly to the **initdb** command. **-o** can be specified multiple times, with all the given options being passed through.

The *initdb-options* should usually be surrounded by single or double quotes to ensure that they are passed through as a group.

-p *path*

Specifies the location of the postgres executable. By default the postgres executable is taken from the same directory as **pg_ctl**, or failing that, the hard-wired installation directory. It is not necessary to use this option unless you are doing something unusual and get errors that the postgres executable was not found.

In init mode, this option analogously specifies the location of the initdb executable.

-s
--silent

Print only errors, no informational messages.

-t *seconds*
--timeout=*seconds*

Specifies the maximum number of seconds to wait when waiting for an operation to complete (see option **-w**). Defaults to the value of the **PGCTLTIMEOUT** environment variable or, if not set, to 60 seconds.

-V
--version

Print the pg_ctl version and exit.

-w
--wait

Wait for the operation to complete. This is supported for the modes start, stop, restart, promote, and register, and is the default for those modes.

When waiting, **pg_ctl** repeatedly checks the server's PID file, sleeping for a short amount of time between checks. Startup is considered complete when the PID file indicates that the server is ready to accept connections. Shutdown is considered complete when the server removes the PID file. **pg_ctl** returns an exit code based on the success of the startup or shutdown.

If the operation does not complete within the timeout (see option **-t**), then **pg_ctl** exits with a nonzero exit status. But note that the operation might continue in the background and eventually succeed.

-W
--no-wait

Do not wait for the operation to complete. This is the opposite of the option **-w**.

If waiting is disabled, the requested action is triggered, but there is no feedback about its success. In that case, the server log file or an external monitoring system would have to be used to check the progress and success of the operation.

In prior releases of PostgreSQL, this was the default except for the stop mode.

-?
--help

Show help about pg_ctl command line arguments, and exit.

If an option is specified that is valid, but not relevant to the selected operating mode, pg_ctl ignores it.

Options for Windows

-e *source*

Name of the event source for pg_ctl to use for logging to the event log when running as a Windows service. The default is PostgreSQL. Note that this only controls messages sent from pg_ctl itself; once started, the server will use the event source specified by its event_source parameter. Should the server fail very early in startup, before that parameter has been set, it might also log using the default event source name PostgreSQL.

-N *servicename*

Name of the system service to register. This name will be used as both the service name and the display name. The default is PostgreSQL.

-P *password*

Password for the user to run the service as.

-S *start-type*

Start type of the system service. *start-type* can be auto, or demand, or the first letter of one of these two. If this option is omitted, auto is the default.

-U *username*

User name for the user to run the service as. For domain users, use the format DOMAIN\username.

ENVIRONMENT

PGCTLTIMEOUT

Default limit on the number of seconds to wait when waiting for startup or shutdown to complete. If not set, the default is 60 seconds.

PGDATA

Default data directory location.

Most **pg_ctl** modes require knowing the data directory location; therefore, the **-D** option is required unless **PGDATA** is set.

pg_ctl, like most other PostgreSQL utilities, also uses the environment variables supported by libpq (see Section 34.15).

For additional variables that affect the server, see **postgres**(1).

FILES

postmaster.pid

pg_ctl examines this file in the data directory to determine whether the server is currently running.

postmaster.opts

If this file exists in the data directory, pg_ctl (in **restart** mode) will pass the contents of the file as options to postgres, unless overridden by the **-o** option. The contents of this file are also displayed in **status** mode.

EXAMPLES

Starting the Server

To start the server, waiting until the server is accepting connections:

```
$ pg_ctl start
```

To start the server using port 5433, and running without **fsync**, use:

```
$ pg_ctl -o "-F -p 5433" start
```

Stopping the Server

To stop the server, use:

```
$ pg_ctl stop
```

The **-m** option allows control over *how* the server shuts down:

```
$ pg_ctl stop -m smart
```

Restarting the Server

Restarting the server is almost equivalent to stopping the server and starting it again, except that by default, **pg_ctl** saves and reuses the command line options that were passed to the previously-running instance. To restart the server using the same options as before, use:

```
$ pg_ctl restart
```

But if **-o** is specified, that replaces any previous options. To restart using port 5433, disabling **fsync** upon restart:

```
$ pg_ctl -o "-F -p 5433" restart
```

Showing the Server Status

Here is sample status output from pg_ctl:

```
$ pg_ctl status

pg_ctl: server is running (PID: 13718)
/usr/local/pgsql/bin/postgres "-D" "/usr/local/pgsql/data" "-p" "5433" "-B" "128"
```

The second line is the command that would be invoked in restart mode.

SEE ALSO

initdb(1), **postgres**(1)