

# Базовый инструментарий Конфигурирование сервера



## **Авторские права**

© Postgres Professional, 2015–2022

Авторы: Егор Рогов, Павел Лузанов, Илья Баштанов

## **Использование материалов курса**

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

## **Обратная связь**

Отзывы, замечания и предложения направляйте по адресу:  
[edu@postgrespro.ru](mailto:edu@postgrespro.ru)

## **Отказ от ответственности**

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Параметры конфигурации

Файлы конфигурации

Управление параметрами на уровнях экземпляра и сеанса

## Задача

управление работой и поведением СУБД

## Установка

для экземпляра — файлы конфигурации

для отдельной базы или пользователя

для текущего сеанса

В PostgreSQL существует большое количество параметров, влияющих на работу СУБД. Параметры позволяют управлять потреблением ресурсов, настраивать работу серверных процессов и многое другое.

Например, при помощи параметра *max\_connections* можно ограничить количество одновременных подключений к серверу.

Полный список и описание параметров конфигурации:

<https://postgrespro.ru/docs/postgresql/13/runtime-config.html>

В этой теме мы не изучаем назначение отдельных параметров конфигурации, а лишь рассматриваем, какими способами им можно устанавливать значения.

Для установки параметров, в первую очередь, используются файлы конфигурации. Если не определено иное, значения, установленные в этих файлах, действуют для всего экземпляра СУБД.

Ряд параметров можно установить для сеансов в отдельной базе данных или для сеансов отдельного пользователя. Такие установки будут иметь предпочтение перед файлами конфигурации. Мы подробнее коснемся этого варианта в следующих темах курса.

Наконец, многими параметрами можно управлять на уровне отдельного сеанса, прямо во время работы.


## Основной файл конфигурации

считывается при старте сервера

 по умолчанию находится в каталоге с данными (PGDATA)

 /etc/postgresql/13/main

## При изменении параметров необходимо перечитать файл

 \$ pg\_ctl reload

 \$ pg\_ctlcluster 13 main reload

=> SELECT pg\_reload\_conf();

изменение ряда параметров требует перезапуска сервера

Основной конфигурационный файл — postgresql.conf.

Расположение файла задается при сборке PostgreSQL. По умолчанию файл располагается в каталоге с данными (PGDATA), но пакетные дистрибутивы обычно размещают этот файл в другом месте, в соответствии с правилами, принятыми в конкретной ОС.

Это текстовый, хорошо документированный, файл, хранящий параметры в формате «ключ=значение».

Если один и тот же параметр указан в конфигурационных файлах несколько раз, использоваться будет значение, считанное последним.

Для вступления в силу внесенных в файл изменений необходимо, чтобы сервер перечитал файл. Для некоторых параметров требуется перезагрузка сервера.

## Файл postgresql.conf и представление pg\_file\_settings

Посмотрим небольшой фрагмент конфигурационного файла.

```
=> SHOW config_file;
```

```
      config_file
-----
/etc/postgresql/13/main/postgresql.conf
(1 row)
```

```
=> SELECT pg_read_file('/etc/postgresql/13/main/postgresql.conf', 1516, 860)
\g (tuples_only=on format=unaligned)
```

```
#-----
# FILE LOCATIONS
#-----

# The default values of these variables are driven from the -D command-line
# option or PGDATA environment variable, represented here as ConfigDir.

data_directory = '/var/lib/postgresql/13/main'      # use data in another directory
                                                    # (change requires restart)
hba_file = '/etc/postgresql/13/main/pg_hba.conf'    # host-based authentication file
                                                    # (change requires restart)
ident_file = '/etc/postgresql/13/main/pg_ident.conf' # ident configuration file
                                                    # (change requires restart)

# If external_pid_file is not explicitly set, no extra PID file is written.
external_pid_file = '/var/run/postgresql/13-main.pid' # write an extra PID file
                                                    # (change requires restart)
```

Чтобы увидеть все незакомментированные строки конфигурационного файла, можно обратиться к представлению pg\_file\_settings:

```
=> SELECT sourceline, name, setting, applied
FROM pg_file_settings
WHERE sourcefile LIKE '/etc/postgresql/13/main/postgresql.conf';
```

sourceline	name	setting	applied
42	data_directory	/var/lib/postgresql/13/main	t
44	hba_file	/etc/postgresql/13/main/pg_hba.conf	t
46	ident_file	/etc/postgresql/13/main/pg_ident.conf	t
50	external_pid_file	/var/run/postgresql/13-main.pid	t
64	port	5432	t
65	max_connections	100	t
67	unix_socket_directories	/var/run/postgresql	t
101	ssl	on	t
103	ssl_cert_file	/etc/ssl/certs/ssl-cert-snakeoil.pem	t
105	ssl_key_file	/etc/ssl/private/ssl-cert-snakeoil.key	t
122	shared_buffers	128MB	t
143	dynamic_shared_memory_type	posix	t
229	max_wal_size	1GB	t
230	min_wal_size	80MB	t
530	log_line_prefix	%m [%p] %q%u@%d	t
564	log_timezone	Europe/Moscow	t
570	cluster_name	13/main	t
586	stats_temp_directory	/var/run/postgresql/13-main.pg_stat_tmp	t
679	datestyle	iso, dmy	t
681	timezone	Europe/Moscow	t
695	lc_messages	en_US.UTF-8	t
697	lc_monetary	ru_RU.UTF-8	t
698	lc_numeric	ru_RU.UTF-8	t
699	lc_time	ru_RU.UTF-8	t
702	default_text_search_config	pg_catalog.english	t

(25 rows)

Столбец applied имеет обманчивое название. Если внести изменения в конфигурационный файл, то значение столбца скажет о том, можно ли применить новое значение без перезапуска сервера. Представление pg\_file\_settings показывает лишь содержимое файлов конфигурации; реальные значения параметров могут отличаться.

## Представление pg\_settings

Действующие значения всех параметров доступны в представлении pg\_settings. Вот что в нем содержится, например, для параметра work\_mem:

```
=> SELECT name, setting, unit,
        boot_val, reset_val,
        source, sourcefile, sourceline,
        pending_restart, context
FROM pg_settings
WHERE name = 'work_mem'\gx
```

```
-[ RECORD 1 ]---+-----
name          | work_mem
setting       | 4096
unit          | kB
boot_val      | 4096
reset_val     | 4096
source        | default
sourcefile    |
sourceline    |
pending_restart | f
context       | user
```

Параметр work\_mem определяет объем памяти, который выделяется для выполнения таких операций, как сортировка или хеш-соединение. Его значение влияет на то, сколько памяти может быть выделено для выполнения запросов. Значение по умолчанию не всегда достаточно для запросов, обрабатывающих большие объемы данных. Более подробно о параметре work\_mem можно узнать в курсе «QPT. Оптимизация запросов».

Рассмотрим ключевые столбцы представления pg\_settings:

- name, setting, unit — название и значение параметра;
- boot\_val — значение по умолчанию;
- reset\_val — значение, которое восстановит команда RESET;
- source — источник текущего значения параметра;
- pending\_restart — значение изменено в файле конфигурации, но для применения требуется перезапуск сервера.

Столбец context определяет действия, необходимые для применения параметра. Среди возможных значений:

- internal — изменить нельзя, значение задано при установке;
- postmaster — требуется перезапуск сервера;
- sighup — требуется перечитать файлы конфигурации;
- superuser — суперпользователь может изменить для своего сеанса;
- user — любой пользователь может изменить для своего сеанса.

## Порядок применения строк postgresql.conf

Если один и тот же параметр встречается в файле несколько раз, то устанавливается значение из последней считанной строки.

Запишем в конец postgresql.conf две строки с параметром work\_mem:

```
student$ echo work_mem=12MB | sudo tee -a /etc/postgresql/13/main/postgresql.conf
```

```
work_mem=12MB
```

```
student$ echo work_mem=8MB | sudo tee -a /etc/postgresql/13/main/postgresql.conf
```

```
work_mem=8MB
```

```
=> SELECT sourceline, name, setting, applied
FROM pg_file_settings
WHERE name = 'work_mem';
```

```
sourceline | name | setting | applied
-----+-----+-----+-----
783 | work_mem | 12MB | f
784 | work_mem | 8MB | t
```

```
(2 rows)
```

По значению applied = f для строки с 12MB понятно, что это значение не может быть применено.

Для параметра work\_mem поле context имеет значение user. Значит, его можно менять прямо во время сеанса, и позже мы увидим, как это сделать.

А чтобы изменить значение во всех сеансах, достаточно перечитать файлы конфигурации:

```
=> SELECT pg_reload_conf();
```

```
pg_reload_conf
```

```
-----
```

```
t
```

```
(1 row)
```

Убедимся, что параметр `work_mem` получил значение из последней строки:

```
=> SELECT name, setting, unit,  
       boot_val, reset_val,  
       source, sourcefile, sourceline,  
       pending_restart, context
```

```
FROM pg_settings
```

```
WHERE name = 'work_mem'\gx
```

```
-[ RECORD 1 ]-----+-----
```

name		work_mem
setting		8192
unit		kB
boot_val		4096
reset_val		8192
source		configuration file
sourcefile		/etc/postgresql/13/main/postgresql.conf
sourceline		784
pending_restart		f
context		user

## Файл конфигурации, управляемый командами SQL

ALTER SYSTEM	добавляет или изменяет строку
SET <i>параметр</i> TO <i>значение</i> ;	
ALTER SYSTEM RESET <i>параметр</i> ;	удаляет строку
ALTER SYSTEM RESET ALL;	удаляет все строки
считывается после postgresql.conf	

## Расположение

всегда в каталоге с данными (PGDATA)

## Действия при изменении

аналогично postgresql.conf

Последним считывается файл postgresql.auto.conf. Этот файл всегда располагается в каталоге данных (PGDATA).

Этот файл не следует изменять вручную. Для его редактирования предназначена команда ALTER SYSTEM. По сути, ALTER SYSTEM представляет собой SQL-интерфейс для управления параметрами конфигурации.

Для применения изменений, сделанных командой ALTER SYSTEM, сервер должен перечитать конфигурационные файлы, как и в случае с изменением файла postgresql.conf.

Содержимое обоих файлов (postgresql.conf и postgresql.auto.conf) можно увидеть через представление pg\_file\_settings. А актуальные значения параметров — в представлении pg\_settings.

Более подробная информация о команде ALTER SYSTEM:

<https://postgrespro.ru/docs/postgresql/13/sql-alterssystem.html>



## Команда ALTER SYSTEM и файл postgresql.auto.conf

Для примера установим параметр work\_mem:

```
=> ALTER SYSTEM SET work_mem TO '16mb';
```

```
ERROR:  invalid value for parameter "work_mem": "16mb"
HINT:   Valid units for this parameter are "B", "kB", "MB", "GB", and "TB".
```

Что случилось?

ALTER SYSTEM выполняет проверку на допустимые значения.

```
=> ALTER SYSTEM SET work_mem TO '16MB';
```

ALTER SYSTEM

Вот теперь все правильно.

В результате выполнения команды значение 16MB записано в файл postgresql.auto.conf:

```
=> SELECT pg_read_file('postgresql.auto.conf')
\g (tuples_only=on format=unaligned)

# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
work_mem = '16MB'
```

Но это значение не применено:

```
=> SHOW work_mem;
```

```
work_mem
-----
8MB
(1 row)
```

Чтобы применить изменения для work\_mem, перечитаем файлы конфигурации:

```
=> SELECT pg_reload_conf();
```

```
pg_reload_conf
-----
t
(1 row)
```

```
=> SELECT name, setting, unit,
       boot_val, reset_val,
       source, sourcefile, sourceline,
       pending_restart, context
FROM pg_settings
WHERE name = 'work_mem'\gx

-[ RECORD 1 ]-----+-----
name          | work_mem
setting       | 16384
unit          | kB
boot_val      | 4096
reset_val     | 16384
source        | configuration file
sourcefile    | /var/lib/postgresql/13/main/postgresql.auto.conf
sourceline    | 3
pending_restart | f
context       | user
```

Для удаления строк из postgresql.auto.conf используется команда ALTER SYSTEM RESET:

```
=> ALTER SYSTEM RESET work_mem;
```

ALTER SYSTEM

```
=> SELECT pg_read_file('postgresql.auto.conf')
\g (tuples_only=on format=unaligned)
```

```
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
```

-----

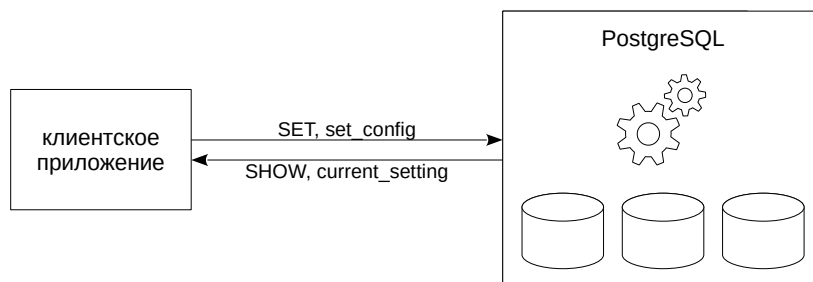
Еще раз перечитаем конфигурацию. Теперь восстановится значение из postgresql.conf:

```
=> SELECT pg_reload_conf();
```

```
pg_reload_conf
-----
t
(1 row)
```

```
=> SELECT name, setting, unit,
       boot_val, reset_val,
       source, sourcefile, sourceline,
       pending_restart, context
FROM pg_settings
WHERE name = 'work_mem'\gx
```

```
-[ RECORD 1 ]---+-----
name          | work_mem
setting       | 8192
unit          | kB
boot_val      | 4096
reset_val     | 8192
source        | configuration file
sourcefile    | /etc/postgresql/13/main/postgresql.conf
sourceline    | 784
pending_restart | f
context       | user
```



установка до конца сеанса или транзакции

установка параметров транзакционна

допускаются пользовательские параметры

Значения параметров можно изменить прямо во время сеанса командой `SET` или функцией `set_config`. А для получения текущего значения служит команда `SHOW` или функция `current_setting`.

Устанавливая новое значение, можно указать срок его действия: до конца сеанса (по умолчанию) или до конца транзакции (`SET LOCAL`).

В любом случае установка параметров транзакционна: в случае отмены текущей транзакции, значения измененных в ней параметров вернутся к состоянию на начало транзакции.

Помимо системных параметров PostgreSQL, этими же командами и функциями можно пользоваться для создания и получения значений пользовательских параметров.

## Установка параметров для текущего сеанса

Для изменения параметров во время сеанса можно использовать команду SET:

```
=> SET work_mem TO '24MB';
```

SET

Или функцию set\_config:

```
=> SELECT set_config('work_mem', '32MB', false);
```

```
set_config
-----
32MB
(1 row)
```

Третий параметр функции говорит о том, нужно ли устанавливать значение только для текущей транзакции (true) или до конца работы сеанса (false). Это важно при работе приложения через пул соединений, когда в одном сеансе могут выполняться транзакции разных пользователей.

---

## Чтение значений параметров во время выполнения

Получить значение параметра можно разными способами:

```
=> SHOW work_mem;
```

```
work_mem
-----
32MB
(1 row)
```

```
=> SELECT current_setting('work_mem');
```

```
current_setting
-----
32MB
(1 row)
```

```
=> SELECT name, setting, unit FROM pg_settings WHERE name = 'work_mem';
```

```
name | setting | unit
-----+-----+-----
work_mem | 32768 | kB
(1 row)
```

---

## Установка параметров внутри транзакции

Откроем транзакцию и установим новое значение work\_mem:

```
=> RESET work_mem;
```

RESET

```
=> BEGIN;
```

BEGIN

```
=> SET work_mem TO '64MB';
```

SET

```
=> SHOW work_mem;
```

```
work_mem
-----
64MB
(1 row)
```

Если транзакция откатывается, установка параметра отменяется, хотя при успешной фиксации новое значение продолжало бы действовать.

```
=> ROLLBACK;
```

ROLLBACK

```
=> SHOW work_mem;
```

```
work_mem
-----
8MB
(1 row)
```

Можно установить значение только до конца текущей транзакции:

```
=> BEGIN;
```

BEGIN

```
=> SET LOCAL work_mem TO '64MB'; -- или set_config('work_mem','64MB',true);
```

SET

```
=> SHOW work_mem;
```

```
work_mem
-----
64MB
(1 row)
```

```
=> COMMIT;
```

COMMIT

По завершении транзакции значение восстанавливается:

```
=> SHOW work_mem;
```

```
work_mem
-----
8MB
(1 row)
```

---

## Пользовательские параметры

Параметры можно создавать прямо во время сеанса, в том числе с предварительной проверкой на существование.

В имени пользовательского параметра обязательно должна быть точка, чтобы отличать их от стандартных параметров.

```
=> SELECT CASE
  WHEN current_setting('myapp.currency_code', true) IS NULL
  THEN set_config('myapp.currency_code', 'RUB', false)
  ELSE
    current_setting('myapp.currency_code')
END;
```

```
current_setting
-----
RUB
(1 row)
```

Теперь myapp.currency\_code можно использовать как глобальную переменную сеанса:

```
=> SELECT current_setting('myapp.currency_code');
```

```
current_setting
-----
RUB
(1 row)
```

Пользовательские параметры можно указать и в postgresql.conf, тогда они автоматически будут инициализироваться во всех сеансах.

Основной файл конфигурации — postgresql.conf

ALTER SYSTEM — SQL-интерфейс для управления параметрами конфигурации в postgresql.auto.conf

При изменениях в файлах нужно перечитать конфигурацию

Многие параметры можно изменять для текущего сеанса

Изменение части параметров требует перезапуска сервера

1. Получите список параметров, для изменения которых требуется перезапуск сервера.
2. В файле `postgresql.conf` сделайте ошибку при изменении параметра `max_connections`.

Перезапустите сервер. Убедитесь, что сервер не стартует, и проверьте журнал сообщений.

Исправьте ошибку и запустите сервер.

2. Расположение файла `postgresql.conf` можно найти, посмотрев значение параметра `config_file`.

Редактируйте файл `postgresql.conf` либо от имени владельца – пользователя `postgres`), либо с правами суперпользователя.

В первом случае удобно открыть новое окно терминала и выполнить в нем команду:

```
sudo su postgres
```

Во втором случае запускайте редактор из командной строки через команду `sudo`, например:

```
sudo vim postgresql.conf
```

## 1. Параметры, изменение которых требует перезапуска сервера

```
=> SELECT name, setting, unit FROM pg_settings WHERE context = 'postmaster';
```

name	setting	unit
archive_mode	off	
autovacuum_freeze_max_age	200000000	
autovacuum_max_workers	3	
autovacuum_multixact_freeze_max_age	400000000	
bonjour	off	
bonjour_name		
cluster_name	13/main	
config_file	/etc/postgresql/13/main/postgresql.conf	
data_directory	/var/lib/postgresql/13/main	
data_sync_retry	off	
dynamic_shared_memory_type	posix	
event_source	PostgreSQL	
external_pid_file	/var/run/postgresql/13-main.pid	
hba_file	/etc/postgresql/13/main/pg_hba.conf	
hot_standby	on	
huge_pages	try	
ident_file	/etc/postgresql/13/main/pg_ident.conf	
ignore_invalid_pages	off	
jit_provider	llvmjit	
listen_addresses	localhost	
logging_collector	off	
max_connections	100	
max_files_per_process	1000	
max_locks_per_transaction	64	
max_logical_replication_workers	4	
max_pred_locks_per_transaction	64	
max_prepared_transactions	0	
max_replication_slots	10	
max_wal_senders	10	
max_worker_processes	8	
old_snapshot_threshold	-1	min
port	5432	
recovery_target		
recovery_target_action	pause	
recovery_target_inclusive	on	
recovery_target_lsn		
recovery_target_name		
recovery_target_time		
recovery_target_timeline	latest	
recovery_target_xid		
restore_command		
shared_buffers	16384	8kB
shared_memory_type	mmap	
shared_preload_libraries		
superuser_reserved_connections	3	
track_activity_query_size	1024	B
track_commit_timestamp	off	
unix_socket_directories	/var/run/postgresql	
unix_socket_group		
unix_socket_permissions	0777	
wal_buffers	512	8kB
wal_level	replica	
wal_log_hints	off	

(53 rows)

## 2. Установка параметра max\_connections

Текущее значение параметра max\_connections:

```
=> SHOW max_connections;
```

```
max_connections
-----
100
(1 row)
```

Допустим, мы решили уменьшить это значение до 50, но ошиблись и вместо нуля написали букву O:

```
student$ echo max_connections=50 | sudo tee -a /etc/postgresql/13/main/postgresql.conf
```

```
max_connections=50
```

Обнаружить ошибку можно, посмотрев в представление pg\_file\_settings:



```
=> SELECT * FROM pg_file_settings WHERE name = 'max_connections'\gx
```

```
-[ RECORD 1 ]-----  
sourcefile | /etc/postgresql/13/main/postgresql.conf  
sourceline | 782  
seqno      | 25  
name       | max_connections  
setting    | 50  
applied    | f  
error      | setting could not be applied
```

Предположим, мы не посмотрели в pg\_file\_settings и все-таки решили перезапустить сервер:

```
=> \q
```

```
student$ sudo pg_ctlcluster 13 main restart
```

Job for postgresql@13-main.service failed because the service did not take the steps required by its unit configuration. See "systemctl status postgresql@13-main.service" and "journalctl -xeu postgresql@13-main.service" for details.

Сервер не запускается. Причина записана в журнал сообщений сервера. Вот последние строки журнала:

```
student$ tail -n 5 /var/log/postgresql/postgresql-13-main.log
```

```
2022-05-06 14:44:53.295 MSK [26083] LOG:  database system is shut down  
2022-05-06 14:44:53.428 MSK [26465] LOG:  invalid value for parameter "max_connections": "50"  
2022-05-06 14:44:53.428 MSK [26465] FATAL:  configuration file "/etc/postgresql/13/main/postgresql.conf" contains errors  
pg_ctl: could not start server  
Examine the log output.
```

Исправим ошибку в файле конфигурации:

```
student$ sudo sed -i 's/50/50/' /etc/postgresql/13/main/postgresql.conf
```

Пробуем запустить сервер:

```
student$ sudo pg_ctlcluster 13 main start
```

Сервер успешно стартовал, проверяем значение max\_connections:

```
student$ /usr/lib/postgresql/13/bin/psql
```

```
=> SHOW max_connections;
```

```
max_connections  
-----  
50  
(1 row)
```

1. Установите параметр `work_mem = 32MB` в командной строке запуска утилиты `psql`.
2. В пакетном дистрибутиве для Ubuntu файл `postgresql.conf` находится не в каталоге `PGDATA`. Каким образом сервер находит этот файл конфигурации при запуске?

1. Используйте один из двух способов: ключ `options` в строке подключения или переменная среды `PGOPTIONS`.

Информация о формировании строки подключения:

<https://postgrespro.ru/docs/postgresql/13/libpq-connect#LIBPQ-CONNSTRING>

2. Расположение файла `postgresql.conf` можно посмотреть в параметре `config_file`.

Чтобы узнать, где задается этот параметр, посмотрите вывод команды `ps` для основного процесса `postgres`. Идентификатор процесса (PID) записан в первой строке файла `postmaster.pid`, который находится в каталоге с данными (`PGDATA`).

## 1. Установка параметров при запуске приложения

Если приложение использует библиотеку libpq для подключения к серверу, установить параметры при запуске можно двумя способами.

Первый способ — указать ключ options в строке параметров подключения:

```
student$ psql "options='-c work_mem=32MB'" -c 'SHOW work_mem'

 work_mem 
-----
 32MB
(1 row)
```

Второй способ — установить переменную окружения PGOPTIONS:

```
student$ export PGOPTIONS='-c work_mem=32MB'; psql -c 'SHOW work_mem'

 work_mem 
-----
 32MB
(1 row)
```

## 2. Где определяется config\_file

Файл postgresql.conf расположен не в каталоге с данными:

```
=> SHOW config_file;

 config_file 
-----
 /etc/postgresql/13/main/postgresql.conf
(1 row)
```

```
=> SHOW data_directory;

 data_directory 
-----
 /var/lib/postgresql/13/main
(1 row)
```

Каким же образом сервер находит postgresql.conf?

В пакетном дистрибутиве для Ubuntu значение параметра config\_file указано в командной строке запуска сервера. Это позволяет перенести postgresql.conf в каталог, отличный от PGDATA.

Команду для запуска сервера можно найти в исходном коде утилиты pg\_ctlcluster, а можно посмотреть в описании процесса postgres. Идентификатор (PID) основного процесса сервера (обычно называемого postmaster), записан в первой строке файла postmaster.pid. Этот файл всегда расположен в каталоге PGDATA.

```
student$ sudo cat /var/lib/postgresql/13/main/postmaster.pid
```

```
26840
/var/lib/postgresql/13/main
1651837501
5432
/var/run/postgresql
localhost
311481      32771
ready
```

```
student$ ps 26840
```

PID	TTY	STAT	TIME	COMMAND
26840	?	Ss	0:00	/usr/lib/postgresql/13/bin/postgres -D /var/lib/postgresql/13/main -c config_file=/etc/postgresql/13/main/postgresql.conf