

# Министерство науки и высшего образования Российской Федерации

---

**Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

*Институт перспективной инженерии*

*Департамент цифровых, робототехнических систем и электроники*

*Межинститутская базовая кафедра*

---

## Отчет по лабораторной работе №1

дисциплины "Администрирование баз данных"

**Выполнил:** Душин Александр Владимирович

4 курс, группа ПИЖ-6-о-22-1

09.03.04 "Программная инженерия"

направленность (профиль) "Разработка и сопровождение программного обеспечения"

очная форма обучения

**Руководитель практики:** Щеголев Алексей Алексеевич

старший преподаватель департамента цифровых, робототехнических систем и электроники института перспективной инженерии

---

## Тема работы

Архитектура СУБД и конфигурация

## Цель работы

Изучить базовые компоненты архитектуры PostgreSQL (процессы, память) и получить практические навыки управления конфигурационными параметрами сервера на разных уровнях (экземпляр, сеанс). Освоить работу с основными и дополнительными файлами конфигурации, а также с представлениями `pg_settings` и `pg_file_settings`.

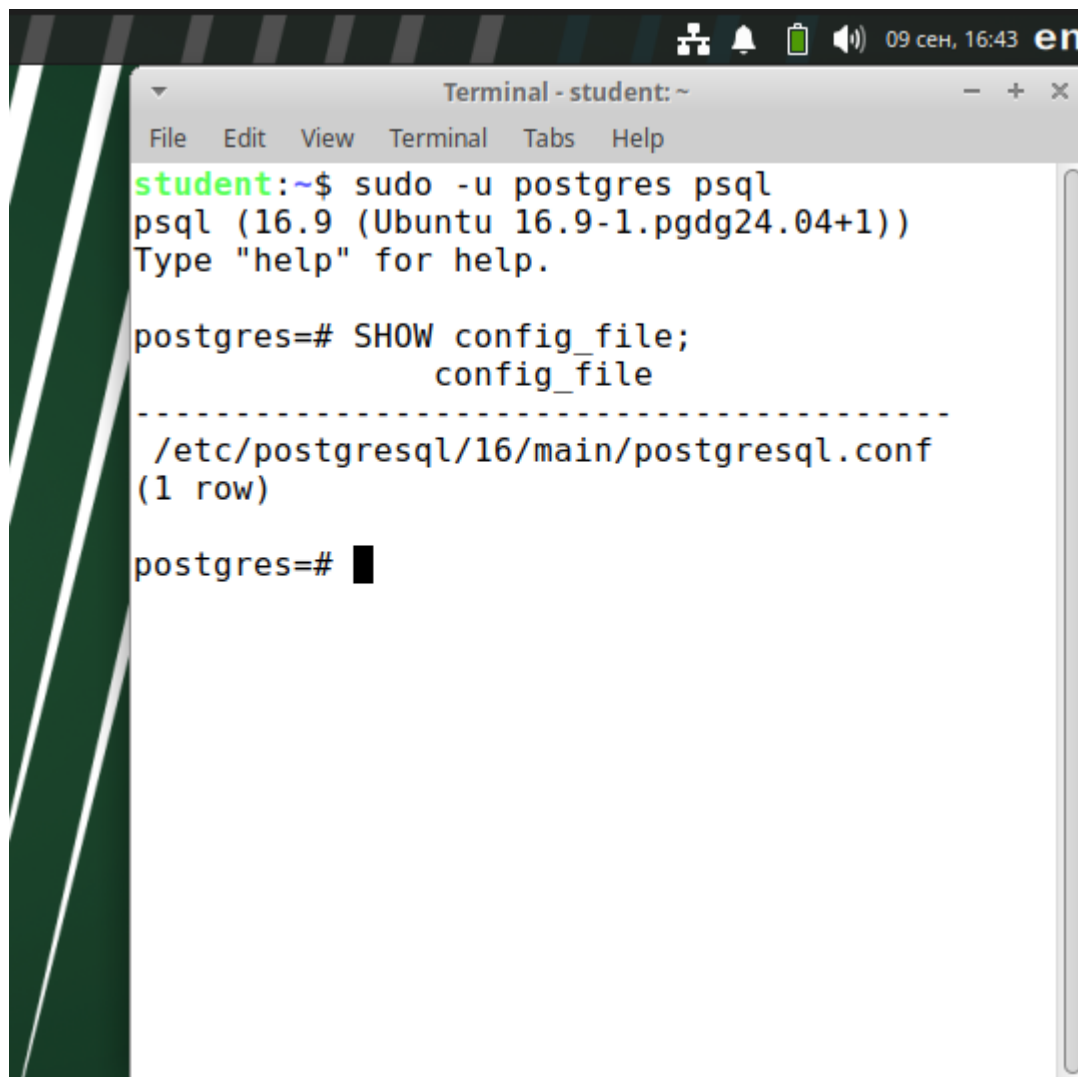
---

## Ход выполнения работы

Часть 1. Исследование параметров и файлов конфигурации

### 1. Определение текущей конфигурации

Выполнено подключение к серверу с помощью `psql`. Определено расположение основного файла конфигурации (`postgresql.conf`) с помощью команды `SHOW config_file;`. (Рис.1)



```
Terminal - student: ~
File Edit View Terminal Tabs Help
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# SHOW config_file;
          config_file
-----
/etc/postgresql/16/main/postgresql.conf
(1 row)

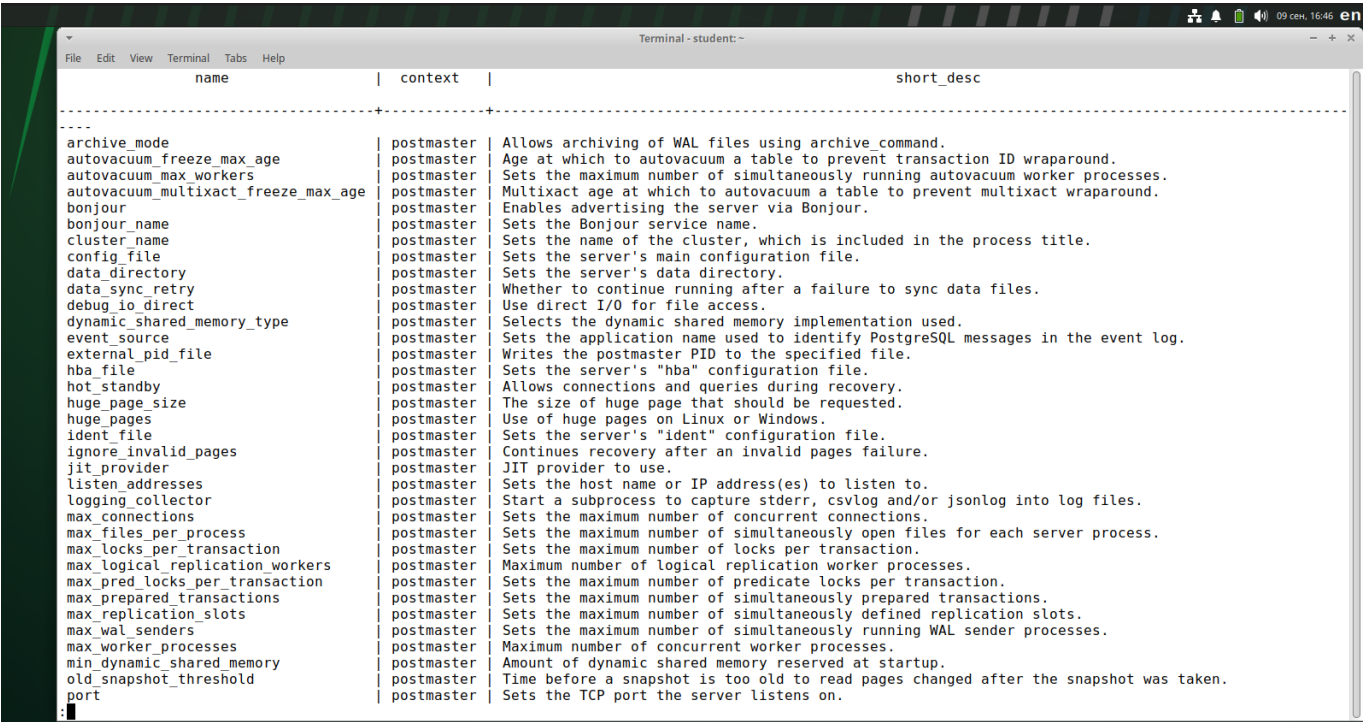
postgres=#
```

Рисунок 1 – Определение расположения конфигурационного файла

## 2. Анализ параметров

Изучено представление `pg_settings`. Найдены параметры, для изменения которых требуется перезагрузка сервера (`context = 'postmaster'`). Найдены параметры с контекстами `sighup` и `user`. (Рис.2-3)

```
SELECT name, context, short_desc
FROM pg_settings
WHERE context = 'postmaster';
```

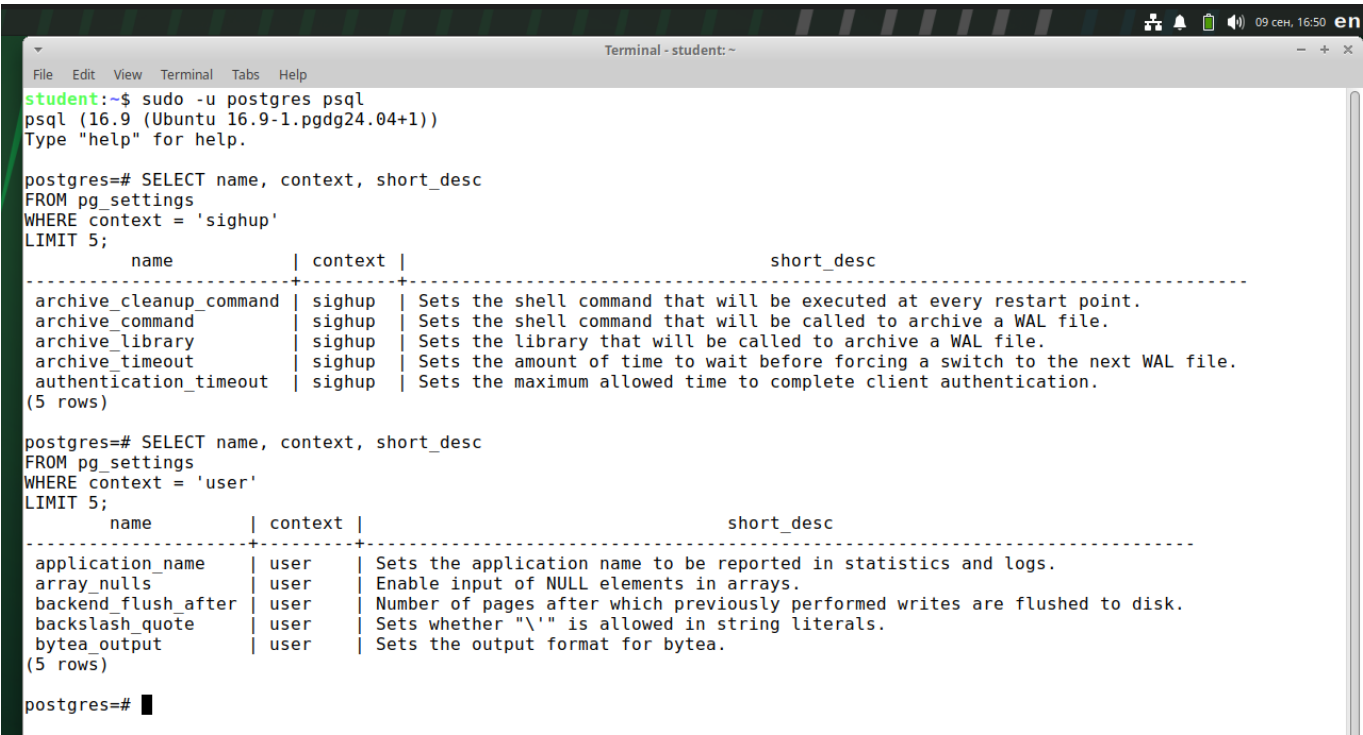


name	context	short_desc
archive_mode	postmaster	Allows archiving of WAL files using archive_command.
autovacuum_freeze_max_age	postmaster	Age at which to autovacuum a table to prevent transaction ID wraparound.
autovacuum_max_workers	postmaster	Sets the maximum number of simultaneously running autovacuum worker processes.
autovacuum_multixact_freeze_max_age	postmaster	Multixact age at which to autovacuum a table to prevent multixact wraparound.
bonjour	postmaster	Enables advertising the server via Bonjour.
bonjour_name	postmaster	Sets the Bonjour service name.
cluster_name	postmaster	Sets the name of the cluster, which is included in the process title.
config_file	postmaster	Sets the server's main configuration file.
data_directory	postmaster	Sets the server's data directory.
data_sync_retry	postmaster	Whether to continue running after a failure to sync data files.
debug_io_direct	postmaster	Use direct I/O for file access.
dynamic_shared_memory_type	postmaster	Selects the dynamic shared memory implementation used.
event_source	postmaster	Sets the application name used to identify PostgreSQL messages in the event log.
external_pid_file	postmaster	Writes the postmaster PID to the specified file.
hba_file	postmaster	Sets the server's "hba" configuration file.
hot_standby	postmaster	Allows connections and queries during recovery.
huge_page_size	postmaster	The size of huge page that should be requested.
huge_pages	postmaster	Use of huge pages on Linux or Windows.
ident_file	postmaster	Sets the server's "ident" configuration file.
ignore_invalid_pages	postmaster	Continues recovery after an invalid pages failure.
jit_provider	postmaster	JIT provider to use.
listen_addresses	postmaster	Sets the host name or IP address(es) to listen to.
logging_collector	postmaster	Start a subprocess to capture stderr, csvlog and/or jsonlog into log files.
max_connections	postmaster	Sets the maximum number of concurrent connections.
max_files_per_process	postmaster	Sets the maximum number of simultaneously open files for each server process.
max_locks_per_transaction	postmaster	Sets the maximum number of locks per transaction.
max_logical_replication_workers	postmaster	Maximum number of logical replication worker processes.
max_pred_locks_per_transaction	postmaster	Sets the maximum number of predicate locks per transaction.
max_prepared_transactions	postmaster	Sets the maximum number of simultaneously prepared transactions.
max_replication_slots	postmaster	Sets the maximum number of simultaneously defined replication slots.
max_wal_senders	postmaster	Sets the maximum number of simultaneously running WAL sender processes.
max_worker_processes	postmaster	Maximum number of concurrent worker processes.
min_dynamic_shared_memory	postmaster	Amount of dynamic shared memory reserved at startup.
old_snapshot_threshold	postmaster	Time before a snapshot is too old to read pages changed after the snapshot was taken.
port	postmaster	Sets the TCP port the server listens on.

Рисунок 2 – Параметры с контекстом postmaster

```
SELECT name, context, short_desc
FROM pg_settings
WHERE context = 'sighup'
LIMIT 5;

SELECT name, context, short_desc
FROM pg_settings
WHERE context = 'user'
LIMIT 5;
```



```
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# SELECT name, context, short_desc
FROM pg_settings
WHERE context = 'sighup'
LIMIT 5;
```

name	context	short_desc
archive_cleanup_command	sighup	Sets the shell command that will be executed at every restart point.
archive_command	sighup	Sets the shell command that will be called to archive a WAL file.
archive_library	sighup	Sets the library that will be called to archive a WAL file.
archive_timeout	sighup	Sets the amount of time to wait before forcing a switch to the next WAL file.
authentication_timeout	sighup	Sets the maximum allowed time to complete client authentication.

(5 rows)

```
postgres=# SELECT name, context, short_desc
FROM pg_settings
WHERE context = 'user'
LIMIT 5;
```

name	context	short_desc
application_name	user	Sets the application name to be reported in statistics and logs.
array_nulls	user	Enable input of NULL elements in arrays.
backend_flush_after	user	Number of pages after which previously performed writes are flushed to disk.
backslash_quote	user	Sets whether "\" is allowed in string literals.
bytea_output	user	Sets the output format for bytea.

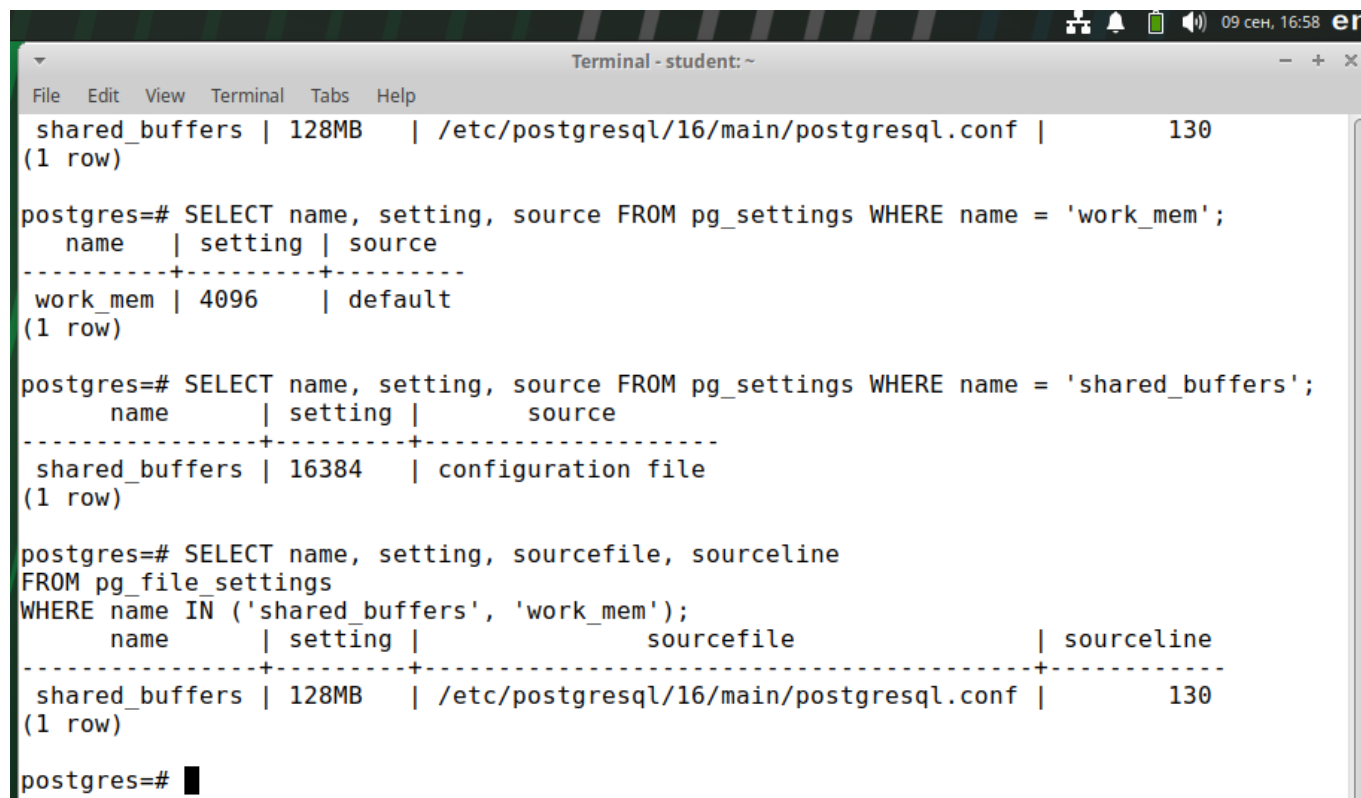
(5 rows)

```
postgres=#
```

Рисунок 3 – Параметры с контекстами sighup и user

### 3. Анализ файлов конфигурации

Изучено представление `pg_file_settings`. Определено, из каких файлов и с какими значениями были считаны текущие настройки параметров `shared_buffers` и `work_mem`. (Рис.4)



```

Terminal - student: ~
File Edit View Terminal Tabs Help

shared_buffers | 128MB      | /etc/postgresql/16/main/postgresql.conf | 130
(1 row)

postgres=# SELECT name, setting, source FROM pg_settings WHERE name = 'work_mem';
   name   | setting | source
-----+-----+-----
work_mem | 4096    | default
(1 row)

postgres=# SELECT name, setting, source FROM pg_settings WHERE name = 'shared_buffers';
   name      | setting | source
-----+-----+-----
shared_buffers | 16384   | configuration file
(1 row)

postgres=# SELECT name, setting, sourcefile, sourceline
FROM pg_file_settings
WHERE name IN ('shared_buffers', 'work_mem');
   name      | setting | sourcefile | sourceline
-----+-----+-----+-----
shared_buffers | 128MB   | /etc/postgresql/16/main/postgresql.conf | 130
(1 row)

postgres=# █

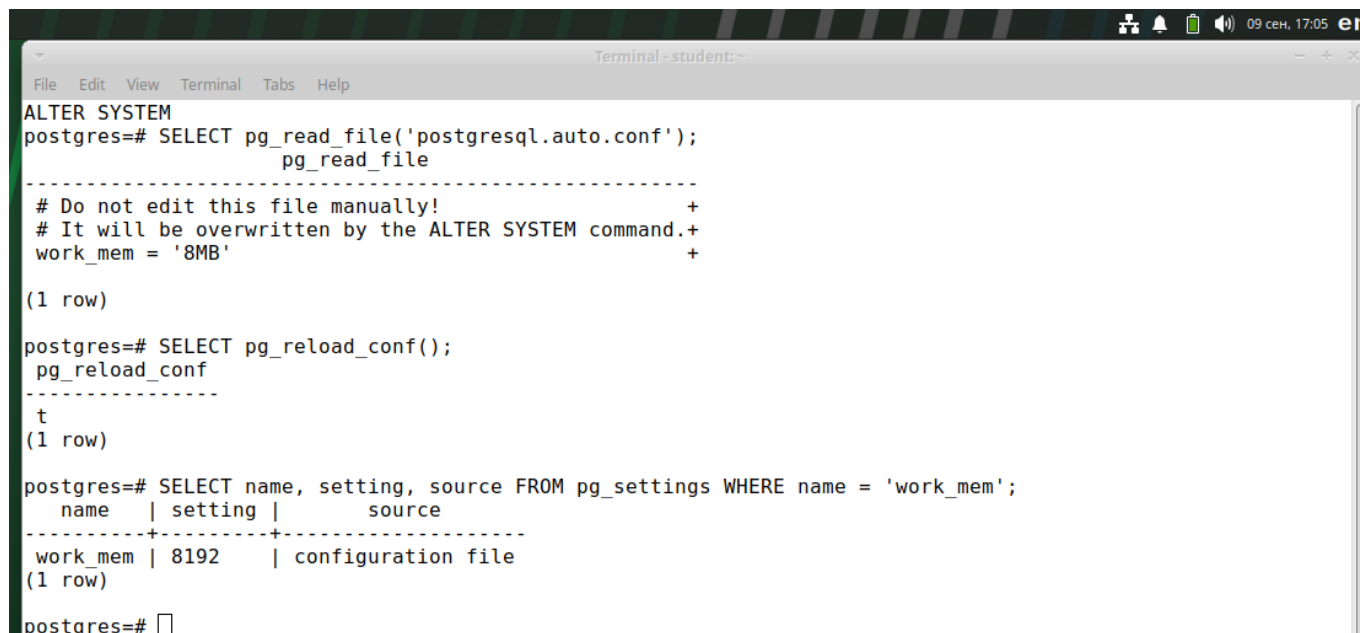
```

Рисунок 4 – Анализ источников параметров конфигурации

## Часть 2. Управление параметрами на уровне экземпляра

### 1. Изменение через ALTER SYSTEM

С использованием команды `ALTER SYSTEM` установлено новое значение для параметра `work_mem`. Проверено, что изменение записалось в файл `postgresql.auto.conf` (используя функцию `pg_read_file`). Применено изменение пересчитыванием конфигурации (`SELECT pg_reload_conf();`). Проверено новое значение параметра и его источник в `pg_settings`. (Рис.5)



```

ALTER SYSTEM
postgres=# SELECT pg_read_file('postgresql.auto.conf');
               pg_read_file
-----
# Do not edit this file manually! +
# It will be overwritten by the ALTER SYSTEM command.+
work_mem = '8MB' +
(1 row)

postgres=# SELECT pg_reload_conf();
 pg_reload_conf
-----
t
(1 row)

postgres=# SELECT name, setting, source FROM pg_settings WHERE name = 'work_mem';
   name   | setting | source
-----+-----+-----
work_mem | 8192    | configuration file
(1 row)

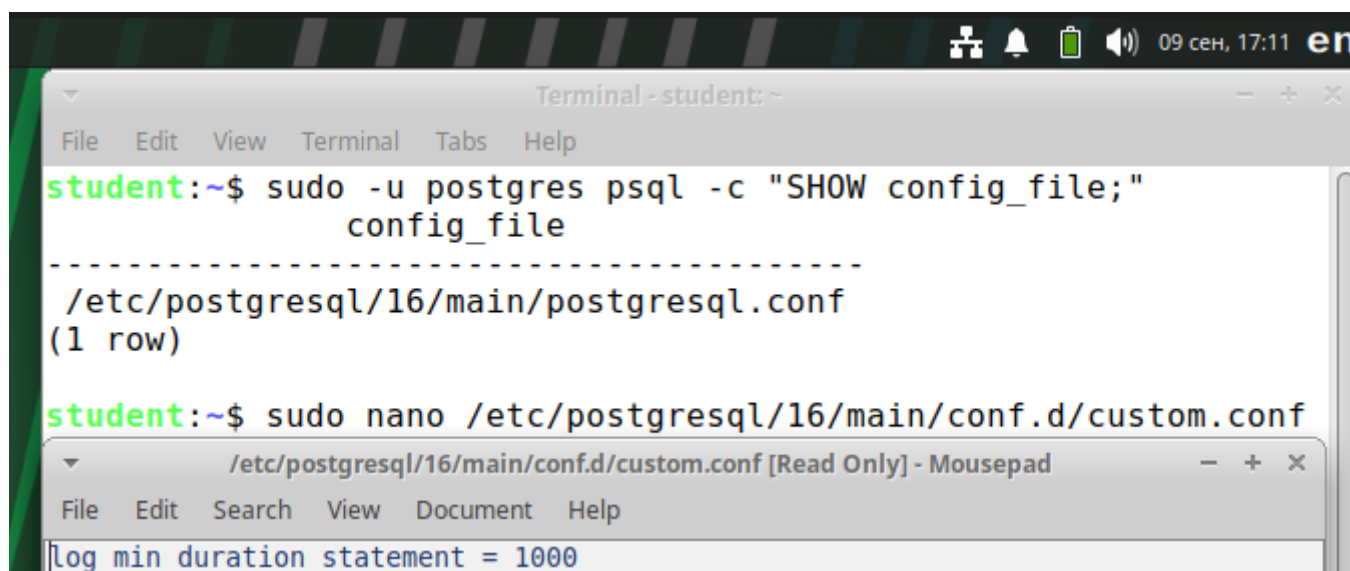
postgres=#

```

Рисунок 5 – Изменение параметра через ALTER SYSTEM

## 2. Изменение через дополнительный файл

Создан файл в каталоге, указанном в директиве `include_dir` основного конфигурационного файла. Установлено значение для параметра `log_min_duration_statement` в этом файле. Применено изменение и проверен результат. (Рис.6-7)



```

student:~$ sudo -u postgres psql -c "SHOW config_file;"
               config_file
-----
/etc/postgresql/16/main/postgresql.conf
(1 row)

student:~$ sudo nano /etc/postgresql/16/main/conf.d/custom.conf

```

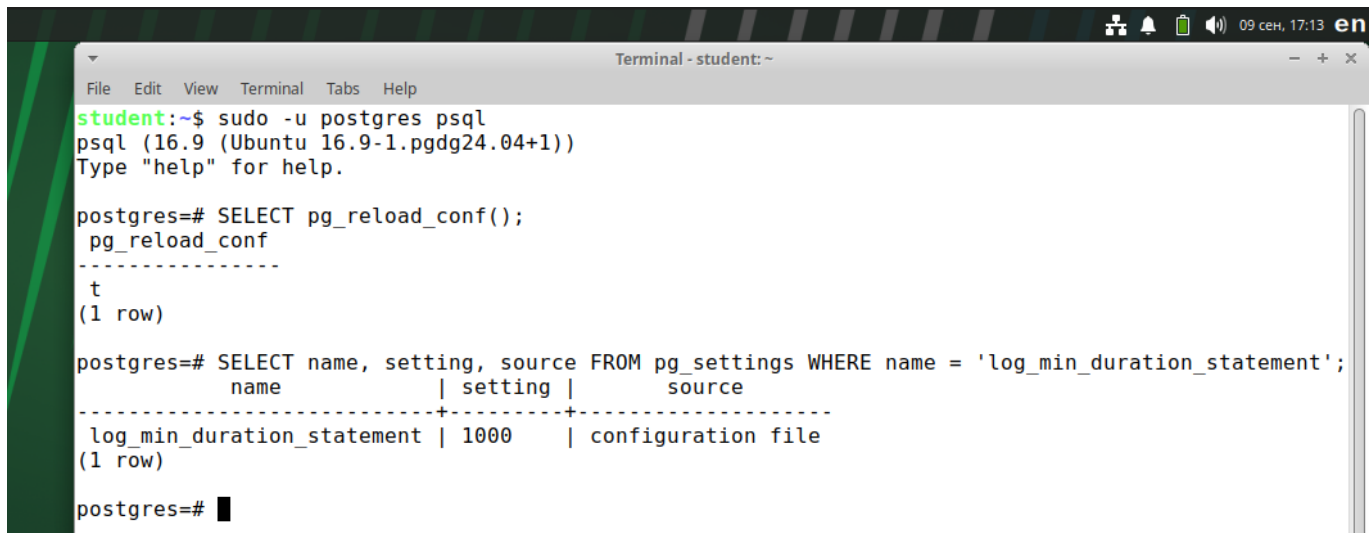
`/etc/postgresql/16/main/conf.d/custom.conf [Read Only] - Mousepad`

```

log_min_duration_statement = 1000

```

Рисунок 6 – Создание дополнительного файла конфигурации



```

student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# SELECT pg_reload_conf();
pg_reload_conf
-----
t
(1 row)

postgres=# SELECT name, setting, source FROM pg_settings WHERE name = 'log_min_duration_statement';
      name      | setting | source
-----+-----+-----
log_min_duration_statement | 1000    | configuration file
(1 row)

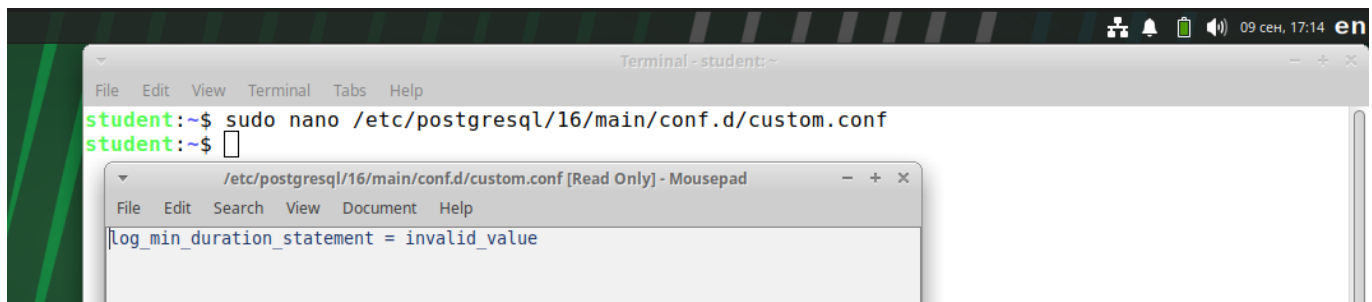
postgres=#

```

Рисунок 7 – Применение изменений из дополнительного файла

### 3. Обработка ошибок в конфигурации

Намеренно внесена синтаксическая ошибка в конфигурационный файл (`invalid_value` вместо числового значения). Выполнена попытка перечитать конфигурацию. Изучено представление `pg_file_settings` для поиска записи об ошибке. Ошибка исправлена и конфигурация перечитана. (Рис.8-11)



```

student:~$ sudo nano /etc/postgresql/16/main/conf.d/custom.conf
student:~$

```

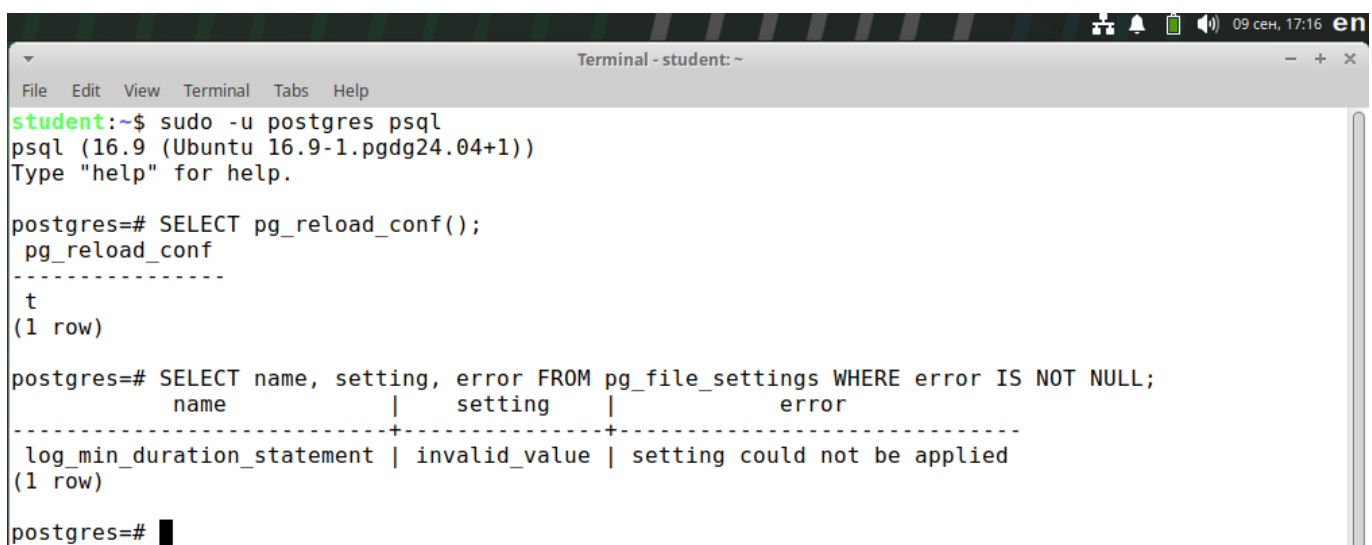
File Edit Search View Document Help

```

log_min_duration_statement = invalid_value

```

Рисунок 8 – Внесение синтаксической ошибки



```

student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# SELECT pg_reload_conf();
pg_reload_conf
-----
t
(1 row)

postgres=# SELECT name, setting, error FROM pg_file_settings WHERE error IS NOT NULL;
      name      | setting | error
-----+-----+-----
log_min_duration_statement | invalid_value | setting could not be applied
(1 row)

postgres=#

```

Рисунок 9 – Обнаружение ошибки в `pg_file_settings`

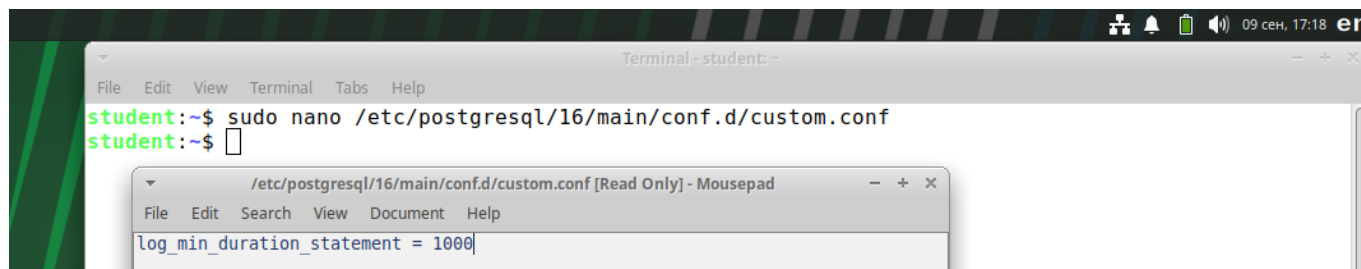


Рисунок 10 – Исправление ошибки в конфигурационном файле

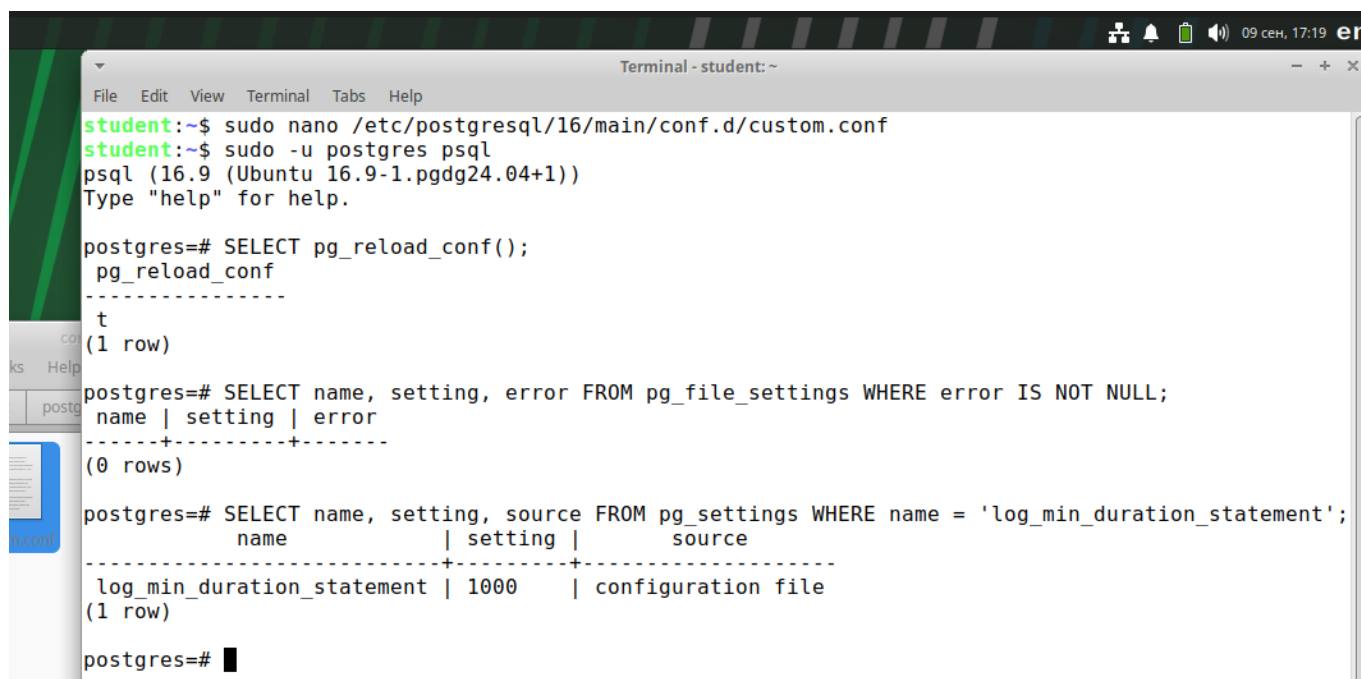
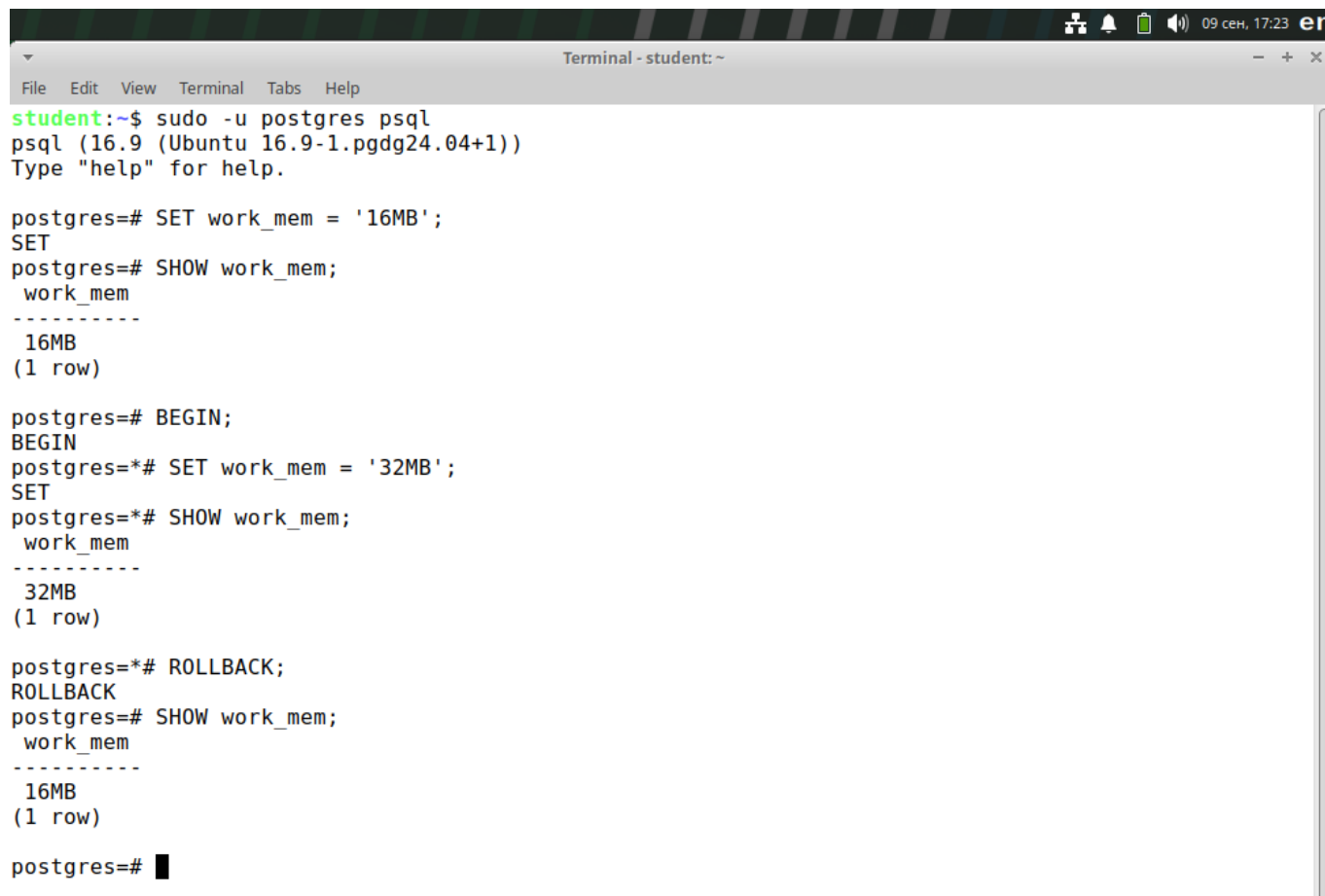


Рисунок 11 – Успешное применение исправленной конфигурации

### Часть 3. Управление параметрами на уровне сеанса

#### 1. Команда SET

В рамках сеанса изменено значение параметра `work_mem` с помощью `SET`. Проверено новое значение. Завершена транзакция с помощью `ROLLBACK` и повторно проверено значение параметра. (Рис.12)

A screenshot of a terminal window titled "Terminal - student: ~". The terminal shows a sequence of PostgreSQL commands and their outputs. The user starts by running 'sudo -u postgres psql', which opens the psql prompt. They then set 'work\_mem' to '16MB' and verify it with 'SHOW work\_mem;', which returns '16MB'. Next, they start a transaction with 'BEGIN', set 'work\_mem' to '32MB', and verify it, which returns '32MB'. Finally, they execute 'ROLLBACK;', and verify 'work\_mem' again, which returns '16MB', demonstrating that the change was not permanent.

```
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# SET work_mem = '16MB';
SET
postgres=# SHOW work_mem;
 work_mem 
-----
 16MB
(1 row)

postgres=# BEGIN;
BEGIN
postgres=# SET work_mem = '32MB';
SET
postgres=# SHOW work_mem;
 work_mem 
-----
 32MB
(1 row)

postgres=# ROLLBACK;
ROLLBACK
postgres=# SHOW work_mem;
 work_mem 
-----
 16MB
(1 row)

postgres=# █
```

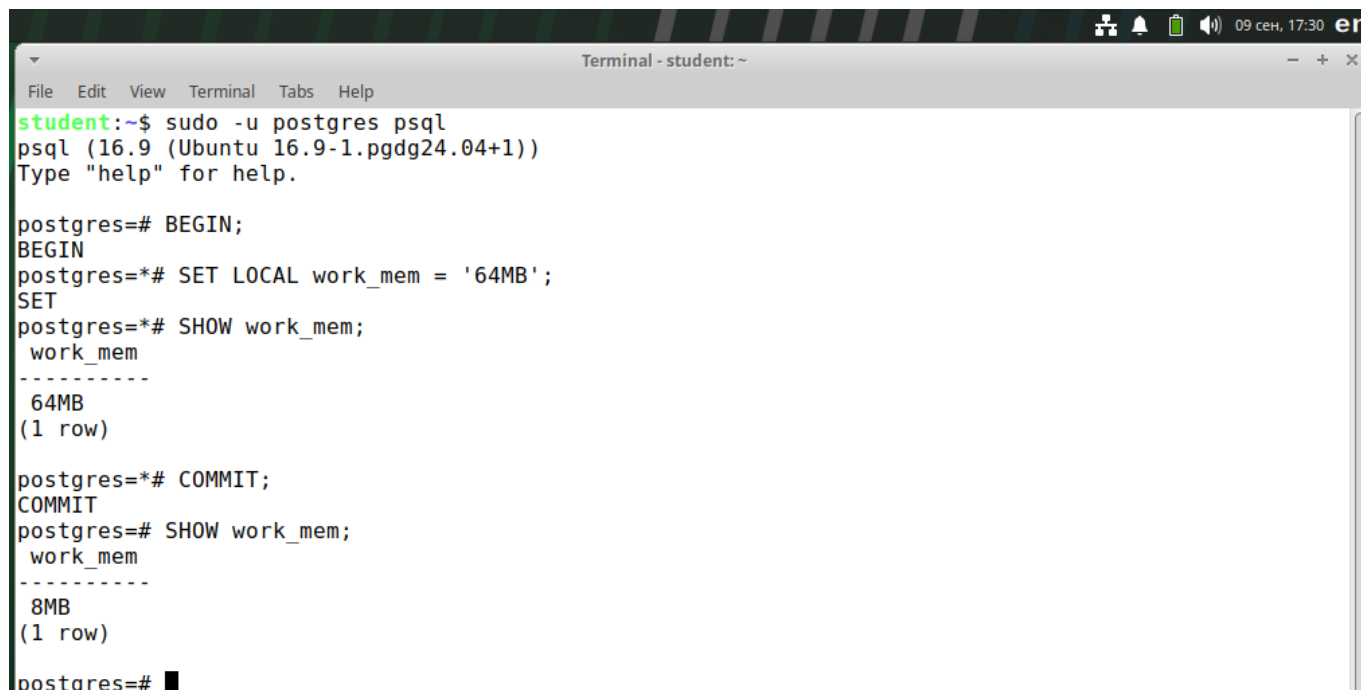
Рисунок 12 – Использование команды SET и влияние ROLLBACK

После выполнения **ROLLBACK** значение **work\_mem** остается **16MB**, а не возвращается к исходному значению. Это происходит потому, что обычная команда **SET** устанавливает параметр на уровне сеанса, а не транзакции. Команда **ROLLBACK** откатывает только изменения данных в рамках транзакции, но не влияет на параметры сеанса, установленные через **SET**.

## 2. Команда SET LOCAL

Открыта транзакция (**BEGIN**). Внутри транзакции использована **SET LOCAL** для изменения **work\_mem**. Проверено изменение. После фиксации транзакции (**COMMIT**) проверено значение параметра снова. (Рис.13)





```
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# BEGIN;
BEGIN
postgres=# SET LOCAL work_mem = '64MB';
SET
postgres=# SHOW work_mem;
 work_mem 
-----
 64MB
(1 row)

postgres=# COMMIT;
COMMIT
postgres=# SHOW work_mem;
 work_mem 
-----
   8MB
(1 row)

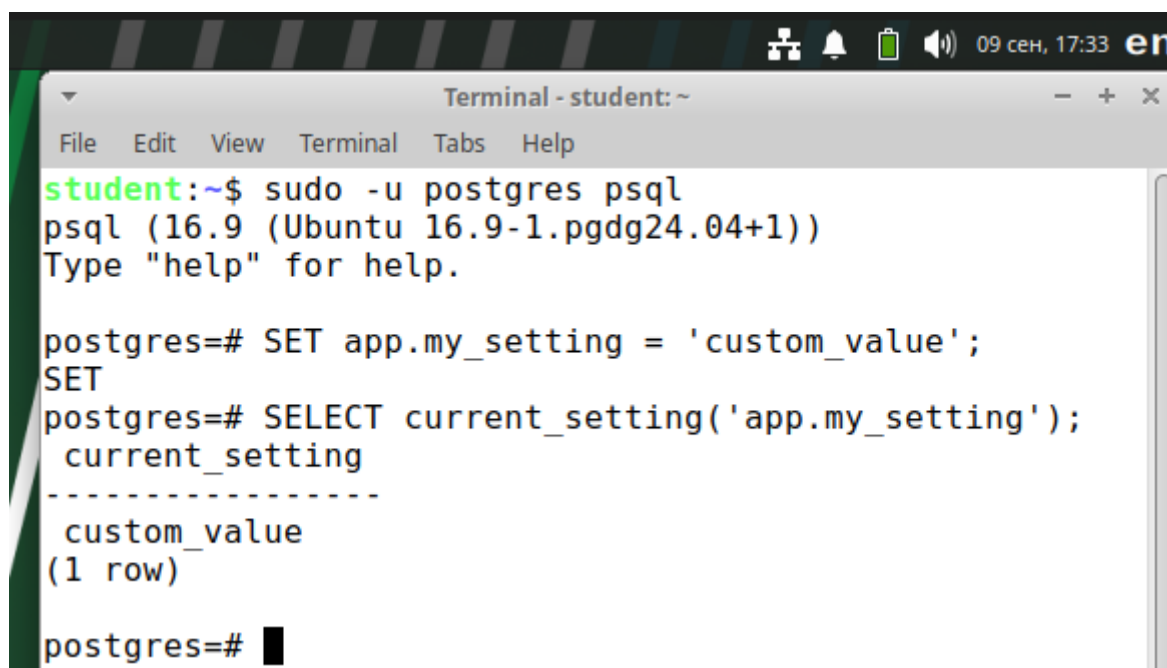
postgres=#
```

Рисунок 13 – Использование SET LOCAL и влияние COMMIT

После выполнения **COMMIT** значение **work\_mem** возвращается к предыдущему значению сеанса (**16MB**). Это ключевое отличие **SET LOCAL** от обычного **SET**: команда **SET LOCAL** устанавливает параметр только в рамках текущей транзакции. При завершении транзакции (как через **COMMIT**, так и через **ROLLBACK**) параметр автоматически возвращается к своему значению на уровне сеанса.

### 3. Пользовательский параметр

Создан и установлено значение для пользовательского параметра (имя содержит точку: **app.my\_setting**). Прочитано его значение с помощью **current\_setting**. (Рис.14)



```
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# SET app.my_setting = 'custom_value';
SET
postgres=# SELECT current_setting('app.my_setting');
 current_setting 
-----
 custom_value
(1 row)

postgres=#
```

Рисунок 14 – Работа с пользовательскими параметрами

## Разница между контекстами параметров

**Postmaster** — параметры, которые могут быть изменены только при перезапуске сервера PostgreSQL. К ним относятся фундаментальные настройки архитектуры, такие как размер разделяемой памяти (`shared_buffers`), максимальное количество подключений (`max_connections`) и порт прослушивания (`port`). Эти параметры определяют базовую конфигурацию кластера и требуют полной перезагрузки процесса postmaster для применения изменений.

**Sighup** — параметры, которые применяются при перечитывании конфигурационных файлов без перезапуска сервера. Изменения вступают в силу после выполнения команды `SELECT pg_reload_conf()` или отправки сигнала SIGHUP процессу postmaster. К таким параметрам относятся настройки логирования (`log_min_duration_statement`), некоторые параметры безопасности и мониторинга. Это позволяет оперативно корректировать поведение сервера без прерывания обслуживания клиентов.

**User** — параметры, которые могут изменяться в рамках пользовательского сеанса командами `SET` и `SET LOCAL`. Примеры включают рабочую память для сортировок (`work_mem`), уровень изоляции транзакций и параметры оптимизатора запросов. Эти изменения действуют только в контексте текущего подключения и не влияют на другие сеансы.

## Разница между способами применения изменений

**ALTER SYSTEM** изменяет параметры на уровне всего кластера PostgreSQL, записывая новые значения в файл `postgresql.auto.conf`. Эти изменения становятся постоянными и применяются ко всем новым подключениям после перечитывания конфигурации или перезапуска сервера (в зависимости от контекста параметра). Команда требует привилегий суперпользователя и предназначена для административного управления конфигурацией.

**SET** устанавливает параметр только в рамках текущего сеанса до его завершения или следующего изменения. Изменения не сохраняются между подключениями и не влияют на другие пользовательские сеансы. Команда может использоваться обычными пользователями для параметров с контекстом user.

**SET LOCAL** действует еще более ограниченно — изменения применяются только в рамках текущей транзакции и автоматически отменяются при её завершении (как при `COMMIT`, так и при `ROLLBACK`). Это полезно для временной настройки параметров для выполнения конкретных операций.

## Процедура поиска и исправления ошибок в конфигурации

При внесении некорректных значений в конфигурационные файлы PostgreSQL не применяет изменения и сохраняет информацию об ошибках в системных представлениях. Для диагностики используется представление `pg_file_settings`, которое показывает все параметры из конфигурационных файлов, включая ошибочные записи.

Процедура поиска ошибки включает выполнение запроса:

```
SELECT name, setting, sourcefile, sourceline, error
FROM pg_file_settings
WHERE error IS NOT NULL;
```

Результат покажет имя параметра, некорректное значение, файл и строку с ошибкой, а также описание проблемы. После выявления и исправления ошибки в соответствующем файле необходимо повторно выполнить `SELECT pg_reload_conf()` для применения корректных настроек.

Такой подход обеспечивает стабильность работы сервера — даже при наличии ошибок в конфигурации PostgreSQL продолжает функционировать с предыдущими корректными настройками, позволяя администратору спокойно исправить проблемы.

---

## Выводы

В ходе выполнения лабораторной работы изучена архитектура PostgreSQL, основанная на модели "процесс на подключение", где главный процесс postmaster управляет кластером и порождает обслуживающие процессы для каждого клиента.

Получены практические навыки управления конфигурационными параметрами на разных уровнях. Освоено использование `ALTER SYSTEM` для изменений на уровне кластера с записью в `postgresql.auto.conf`, команд `SET` и `SET LOCAL` для настройки параметров сеанса и транзакции соответственно. Изучены различия между контекстами параметров: postmaster требует перезапуска сервера, sighup — пересчитывания конфигурации, user позволяет изменения в рамках сеанса.

Освоена работа с представлениями `pg_settings` для анализа текущих параметров и `pg_file_settings` для диагностики конфигурационных файлов и поиска ошибок. Практическая работа с дополнительными конфигурационными файлами через `include_dir` показала гибкость системы управления настройками PostgreSQL.