

---

# Carnival

нояб. 15, 2021



<b>1</b>	<b>Оборудование (Host)</b>	<b>1</b>
<b>2</b>	<b>Шаги (Step)</b>	<b>5</b>
2.1	Работа с контекстом . . . . .	5
2.2	Перегрузка контекста . . . . .	6
<b>3</b>	<b>Задача (Task)</b>	<b>7</b>
3.1	Простые задачи . . . . .	7
3.2	Встроенные задачи . . . . .	8
3.3	Результат выполнения Task.step . . . . .	8
<b>4</b>	<b>Интерфейс командной строки (cli)</b>	<b>9</b>
<b>5</b>	<b>Шаблоны jinja2 (templates)</b>	<b>11</b>
<b>6</b>	<b>Базовые команды (cmd)</b>	<b>13</b>
6.1	Apt . . . . .	13
6.2	Cli . . . . .	14
6.3	Filesystem . . . . .	14
6.4	System . . . . .	15
6.5	Systemd . . . . .	16
6.6	Transfer . . . . .	17
<b>7</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Содержание модулей Python</b>	<b>21</b>
	<b>Алфавитный указатель</b>	<b>23</b>



---

Оборудование (Host)

---

Объект, представляющий единицу оборудования.

Carnival не предоставляет никаких сложных абстракций для работы с группами хостов, подразумевая что вы будете использовать встроенные коллекции python и организуете работу так, как будет удобно для вашей задачи. В простом случае, можно передавать хосты прямо в коде файла *carnival\_tasks.py*.

```
>>> class SetupFrontend(Task):
>>>     def run(self, **kwargs):
>>>         self.step(Frontend(), SSHHost("1.2.3.4", packages=["htop", ]))
```

В более сложных, создать списки в файле *inventory.py*

```
>>> # inventory.py
>>> frontends = [
>>>     LocalHost(),
>>>     SSHHost("1.2.3.5"),
>>> ]
```

```
>>> # carnival_tasks.py
>>> import inventory as i
>>> class SetupFrontend(Task):
>>>     def run(self, **kwargs):
>>>         self.step(Frontend(), i.frontends)
```

`carnival.host.AnyConnection = typing.Union[<sphinx.ext.autodoc.importer._MockObject object>, <sphinx.ext.autodoc.importer._MockObject object>]`  
 Список адресов которые трактуются как локальное соединение .. deprecated:: 1.4

Host is deprecated, use LocalHost or SSHHost explicitly

```
class carnival.host.Host(addr: str, ssh_user: Optional[str] = None, ssh_password: Optional[str] = None, ssh_port: int = 22, ssh_gateway: Optional[SSHHost] = None, ssh_connect_timeout: int = 10, missing_host_key_policy: <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b80c4f0> = <sphinx.ext.autodoc.importer._MockObject object>, **context)
```

Параметры

- `addr` – Адрес сервера для SSH или «local» для локального соединения
- `ssh_user` – Пользователь SSH
- `ssh_password` – Пароль SSH
- `ssh_port` – SSH порт
- `ssh_connect_timeout` – SSH таймаут соединения
- `ssh_gateway` – Gateway
- `context` – Контекст хоста

Не рекомендуется, начиная с версии 1.4: Host is deprecated, use LocalHost or SSHHost explicitly

```
class carnival.host.LocalHost(**context)
```

Локальный хост, работает по локальному терминалу

`host`

Remove user and port parts, return just address

```
class carnival.host.SSHHost(addr: str, ssh_user: Optional[str] = None,
    ssh_password: Optional[str] = None, ssh_port: int = 22,
    ssh_gateway: Optional[SSHHost] = None, ssh_connect_timeout: int = 10,
    missing_host_key_policy: <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b80c4f0> = <sphinx.ext.autodoc.importer._MockObject object>, **context)
```

Удаленный хост, работает по SSH

`is_connection_local()` → bool

Check if host's connection is local

```
host.LOCAL_ADDRS = ['local', 'localhost']
```

```
class carnival.host.Host(addr: str, ssh_user: Optional[str] = None, ssh_password: Optional[str] = None,
    ssh_port: int = 22, ssh_gateway: Optional[SSHHost] = None, ssh_connect_timeout: int = 10,
    missing_host_key_policy: <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b80c4f0> = <sphinx.ext.autodoc.importer._MockObject object>, **context)
```

### Параметры

- `addr` – Адрес сервера для SSH или «local» для локального соединения
- `ssh_user` – Пользователь SSH
- `ssh_password` – Пароль SSH
- `ssh_port` – SSH порт
- `ssh_connect_timeout` – SSH таймаут соединения
- `ssh_gateway` – Gateway
- `context` – Контекст хоста

Не рекомендуется, начиная с версии 1.4: Host is deprecated, use LocalHost or SSHHost explicitly

```
__init__(addr: str, ssh_user: Optional[str] = None, ssh_password: Optional[str] = None,
    ssh_port: int = 22, ssh_gateway: Optional[SSHHost] = None, ssh_connect_timeout: int = 10,
    missing_host_key_policy: <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b80c4f0> = <sphinx.ext.autodoc.importer._MockObject object>, **context)
```

### Параметры

- `addr` – Адрес сервера
- `ssh_user` – Пользователь SSH
- `ssh_password` – Пароль SSH
- `ssh_port` – SSH порт
- `ssh_connect_timeout` – SSH таймаут соединения
- `ssh_gateway` – Gateway
- `context` – Контекст хоста

```
class carnival.host.SSHHost(addr: str, ssh_user: Optional[str] = None,
                             ssh_password: Optional[str] = None, ssh_port: int
                             = 22, ssh_gateway: Optional[SSHHost] = None,
                             ssh_connect_timeout: int = 10, missing_host_key_policy:
                             <sphinx.ext.autodoc.importer._MockObject object at
                             0x7f7f9b80c4f0> = <sphinx.ext.autodoc.importer._MockObject
                             object>, **context)
```

Удаленный хост, работает по SSH

```
__init__(addr: str, ssh_user: Optional[str] = None, ssh_password: Optional[str] = None,
          ssh_port: int = 22, ssh_gateway: Optional[SSHHost] = None, ssh_connect_timeout:
          int = 10, missing_host_key_policy: <sphinx.ext.autodoc.importer._MockObject object
          at 0x7f7f9b80c4f0> = <sphinx.ext.autodoc.importer._MockObject object>, **context)
```

#### Параметры

- `addr` – Адрес сервера
- `ssh_user` – Пользователь SSH
- `ssh_password` – Пароль SSH
- `ssh_port` – SSH порт
- `ssh_connect_timeout` – SSH таймаут соединения
- `ssh_gateway` – Gateway
- `context` – Контекст хоста

```
is_connection_local() → bool
    Check if host's connection is local
```

```
class carnival.host.LocalHost(**context)
    Локальный хост, работает по локальному терминалу
```

```
__init__(**context) → None
    Initialize self. See help(type(self)) for accurate signature.
```

```
host
    Remove user and port parts, return just address
```





```
class carnival.Step(**context)
```

Объект, предназначенный для выполнения группы команд с какой-то целью. Вызывается из класса *carnival.Task* для выполнения команд (*carnival.cmd*) на определенных хостах.

Может требовать наличие определенных контекстных переменных для работы, указав их в аргументах метода *run*. Может вернуть значение для дальнейшего использования.

В следующем примере переменная *disk\_name* будет передана в *run*, а *install* пропущена.

```
>>> host = Host(
>>>     # Адрес
>>>     "1.2.3.4",
>>>
>>>     # Контекст хоста
>>>     disk_name="/dev/sda1", install=['nginx', 'http', ]
>>> )
>>> ...
>>> class DiskUsage(Step):
>>>     def run(self, disk_name: str):
>>>         ...
```

```
__init__(**context)
```

**Параметры context** – Переменные контекста, назначенные при вызове Шага

*run(\*\*kwargs)* → None

Метод который нужно определить для выполнения команд

**Параметры kwargs** – Автоматические подставляемые переменные контекста, под-  
держивается *\*\*kwargs*

## 2.1 Работа с контекстом

Существует несколько видов контекста:

- Специальные переменные контекста
- Переменные окружения, начинающиеся с префикса `CARNIVAL_CTX_`. Переменную `CARNIVAL_CTX_WORKDIR` можно получить по имени `WORKDIR`. Поддерживается `.env-файлы`.
- Контекст хоста (`Host.__init__(..., **context)`).
- Контекст шага (`Step.__init__(**context)`).

#### Специальные переменные контекста:

- `host` - host, на котором вызвано выполнение.

## 2.2 Перегрузка контекста

Список необходимых переменных контекста для шага строится автоматически из сигнатуры метода `Step.run` из переменных контекста хоста и шага, поддерживается `**kwargs`.

- Специальные переменные контекста имеют самый низкий приоритет. Если существует переменная контекста хоста с таким же именем - она будет иметь более высокий приоритет, и перезапишет значение в аргументе `Step.run`.
- Переменные окружения имеют приоритет выше, чем специальные переменные.
- Переменные контекста шага имеют самый высокий приоритет, они перезаписывают любые другие переменные.

```
>>> host = Host("1.2.3.4", disk_name="/dev/sda1", install=['nginx', 'htop', ])
>>>
>>> class DiskUsage(Step):
>>>     def run(self, disk_name: str):
>>>         return cmd.cli.run(f"du -h {disk_name}")
>>>
>>> class FrontendDiskUsage(Task):
>>>     def run(self):
>>>         # получить использование диска `/dev/sda1` (`host.disk_name`)
>>>         self.step(DiskUsage(), host)
>>>
>>>         # получить использование диска `/dev/sda2`
>>>         # (перезгружается из контекста шага)
>>>         self.step(DiskUsage(disk_name="/dev/sda2"), host)
>>>
```

## Задача (Task)

```
class carnival.Task(dry_run: bool)
```

Задача это единица выполнения одного или нескольких шагов на определенных хостах.

Именование задач.

Полное имя задачи состоит из двух частей. `<module_name>.<name>`. `carnival` автоматически генерирует имена задач из этих частей, но есть возможность управлять этим вручную, используя два атрибута класса `Task`.

`name`: название задачи. если не определено имя будет сгенерировано автоматически.  
`module_name`: имя модуля. если назначить пустую строку, полное имя будет включать только название задачи.

```
call_task(task_class: Type[Task]) → Any
```

Запустить другую задачу Возвращает результат работы задачи

```
run() → Any
```

Реализация выполнения задачи

```
step(steps: Union[carnival.step.Step, List[carnival.step.Step]], hosts:
      Union[carnival.host.LocalHost, carnival.host.SSHHost, List[Union[carnival.host.LocalHost,
      carnival.host.SSHHost]]]) → List[carnival.task.TaskResult]
```

Запустить шаг(и) на хост(ах) Возвращает объект `TaskResult` для получения результатов работы каждого шага на каждом хосте

### 3.1 Простые задачи

```
class carnival.SimpleTask(dry_run: bool)
```

Запустить шаги `self.steps` на хостах `self.hosts`

## 3.2 Встроенные задачи

carnival имеет встроенные задачи для удобства использования

```
class carnival.internal_tasks.Help(dry_run: bool)
    Показать список доступных задач

    help = 'List available commands and help'

    module_name = ''

    run() → None
        Реализация выполнения задачи
```

## 3.3 Результат выполнения Task.step

```
class carnival.task.TaskResult(host: Union[carnival.host.LocalHost, carnival.host.SSHHost],
                               step: carnival.step.Step, result: Any)
    Возвращается вызовом метода Task.step
```

---

Интерфейс командной строки (cli)

---

`carnival.cli.except_hook(type: Type[Any], value: Any, traceback: Any) → None`

`carnival.cli.is_completion_script(complete_var: str) → bool`

`carnival.cli.main() → int`

```
>>> $ poetry run python -m carnival --help
>>> Usage: python -m carnival [OPTIONS] {help|test}...
>>> Options:
>>> -d, --dry_run  Simulate run
>>> --debug        Turn on debug mode
>>> --help        Show this message and exit.
```



---

## Шаблоны jinja2 (templates)

---

```
carnival.templates.render(template_path: str, **context) → str
```





---

## Базовые команды (cmd)

---

Модуль `carnival.cmd` содержит базовые команды для взаимодействия с сервером. Его цель - оставаться простым и помогать в написании шагов (Step).

Для написания сложных сценариев предполагается использовать шаги(Step).

Основные шаги доступны в отдельном репозитории: [<https://github.com/carnival-org/carnival-contrib>](https://github.com/carnival-org/carnival-contrib).

### 6.1 Apt

```
carnival.cmd.apt.force_install(pkgname: str, version: Optional[str] = None, update: bool = False,
                               hide: bool = False) → None
```

Установить пакет без проверки установлен ли он

```
carnival.cmd.apt.get_installed_version(pkgname: str) → Optional[str]
```

Получить установленную версию пакета

**Результат** Версия пакета если установлен, *None* если пакет не установлен

```
carnival.cmd.apt.get_pkg_versions(pkgname: str) → List[str]
```

Получить список доступных версий пакета

```
carnival.cmd.apt.install(pkgname: str, version: Optional[str] = None, update: bool = True, hide:
                          bool = False) → bool
```

Установить пакет если он еще не установлен в системе

#### Параметры

- `pkgname` – название пакета
- `version` – версия
- `update` – запустить apt-get update перед установкой
- `hide` – скрыть вывод этапов

**Результат** *True* если пакет был установлен, *False* если пакет уже был установлен ранее

`carnival.cmd.apr.install_multiple(*pkg_names, update: bool = True, hide: bool = False) → bool`  
Установить несколько пакетов, если они не установлены

#### Параметры

- `pkg_names` – список пакетов которые нужно установить
- `update` – запустить apt-get update перед установкой
- `hide` – скрыть вывод этапов

**Результат** `True` если хотя бы один пакет был установлен, `False` если все пакеты уже были установлен ранее

`carnival.cmd.apr.is_pkg_installed(pkgname: str, version: Optional[str] = None) → bool`  
Проверить установлен ли пакет Если версия не указана - проверяется любая

`carnival.cmd.apr.remove(*pkg_names, hide: bool = False) → None`  
Удалить пакет

#### Параметры

- `pkg_names` – список пакетов которые нужно удалить
- `hide` – скрыть вывод этапов

## 6.2 Cli

`carnival.cmd.cli.ptty(command: str, **kwargs) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b7a6640>`  
Запустить команду, используя псевдотерминальную сессию

См <https://docs.pyinvoke.org/en/latest/api/runners.html>

`carnival.cmd.cli.run(command: str, **kwargs) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b7a6640>`  
Запустить команду

## 6.3 Filesystem

`carnival.cmd.fs.ensure_dir_exists(path: str, user: Optional[str] = None, group: Optional[str] = None, mode: Optional[str] = None) → None`

Проверить что директория существует и параметры соответствуют заданным

<https://fabric-patchwork.readthedocs.io/en/latest/api/files.html#patchwork.files.directory>

#### Параметры

- `path` – путь до директории
- `user` – владелец
- `group` – группа
- `mode` – права

`carnival.cmd.fs.is_dir_exists(dir_path: str) → bool`  
Узнать существует ли директория

**Параметры** `dir_path` – путь до директории

`carnival.cmd.fs.is_file_contains(filename: str, text: str, exact: bool = False, escape: bool = True)`

→ bool  
Содержит ли файл текст См [<https://fabric-patchwork.readthedocs.io/en/latest/api/files.html#patchwork.files.contains>](https://fabric-patchwork.readthedocs.io/en/latest/api/files.html#patchwork.files.contains)

#### Параметры

- `filename` – путь до файла
- `text` – текст который нужно искать
- `exact` – точное совпадение
- `escape` – экранировать ли текст

`carnival.cmd.fs.is_file_exists(path: str) → bool`

Проверить существует ли файл [<https://fabric-patchwork.readthedocs.io/en/latest/api/files.html#patchwork.files.exists>](https://fabric-patchwork.readthedocs.io/en/latest/api/files.html#patchwork.files.exists)

**Параметры** `path` – путь до файла

`carnival.cmd.fs.mkdirs(*dirs) → List[<sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b7b3fd0>]`

Создать директории

**Параметры** `dirs` – пути которые нужно создать

## 6.4 System

`carnival.cmd.system.get_current_user_id() → int`

Получить id текущего пользователя

`carnival.cmd.system.get_current_user_name() → str`

Получить имя текущего пользователя

`carnival.cmd.system.is_current_user_root() → bool`

Проверить что текущий пользователь - *root*

`carnival.cmd.system.set_password(username: str, password: str) →`  
`<sphinx.ext.autodoc.importer._MockObject object at`  
`0x7f7f9b7a6be0>`

Установить пароль пользователю

#### Параметры

- `username` – Пользователь
- `password` – Новый пароль

`carnival.cmd.system.ssh_authorized_keys_add(ssh_key: str, keys_file: str =`  
`'ssh/authorized_keys') → bool`

Добавить ssh ключ в *authorized\_keys*

#### Параметры

- `ssh_key` – ключ
- `keys_file` – путь до файла *authorized\_keys*

**Результат** *True* если ключ был добавлен, *False* если ключ уже был в файле

`carnival.cmd.system.ssh_authorized_keys_ensure(*ssh_keys) → List[bool]`

Добавить несколько ssh-ключей в авторизованные

**Параметры** `ssh_keys` – ssh-ключи

**Результат** Список *True* если ключ был добавлен, *False* если ключ уже был в файле

```
carnival.cmd.system.ssh_authorized_keys_list() → List[str]
```

Получить список авторизованных ssh-ключей сервера

```
carnival.cmd.system.ssh_copy_id(pubkey_file: str = '~/.ssh/id_rsa.pub') → bool
```

Добавить публичный ssh-ключ текущего пользователя в авторизованные

**Параметры** `pubkey_file` – путь до файла с публичным ключем

**Результат** *True* если ключ был добавлен, *False* если ключ уже был в файле

## 6.5 Systemd

```
carnival.cmd.systemd.daemon_reload() → <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b7a69a0>
```

Перезагрузить systemd

```
carnival.cmd.systemd.disable(service_name: str, reload_daemon: bool = False, stop_now: bool = True) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b7a69a0>
```

Убрать сервис из автозапуска

**Параметры**

- `service_name` – имя сервиса
- `reload_daemon` – перезагрузить systemd
- `stop_now` – Остановить сервис

```
carnival.cmd.systemd.enable(service_name: str, reload_daemon: bool = False, start_now: bool = True) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b7a69a0>
```

Добавить сервис в автозапуск

**Параметры**

- `service_name` – имя сервиса
- `reload_daemon` – перезагрузить systemd
- `start_now` – запустить сервис после добавления

```
carnival.cmd.systemd.restart(service_name: str) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b7a69a0>
```

Перезапустить сервис

**Параметры** `service_name` – имя сервиса

```
carnival.cmd.systemd.start(service_name: str, reload_daemon: bool = False) → <sphinx.ext.autodoc.importer._MockObject object at 0x7f7f9b7a69a0>
```

Запустить сервис

**Параметры**

- `service_name` – имя сервиса
- `reload_daemon` – перезагрузить systemd

```
carnival.cmd.systemd.stop(service_name: str, reload_daemon: bool = False) →
    <sphinx.ext.autodoc.importer._MockObject object at
    0x7f7f9b7a69a0>
```

Остановить сервис

#### Параметры

- `service_name` – имя сервиса
- `reload_daemon` – перезагрузить systemd

## 6.6 Transfer

```
carnival.cmd.transfer.get(remote: str, local: str, preserve_mode: bool = True)
    → <sphinx.ext.autodoc.importer._MockObject object at
    0x7f7f9b7b3850>
```

Скачать файл с сервера <<http://docs.fabfile.org/en/2.5/api/transfer.html#fabric.transfer.Transfer.get>>

#### Параметры

- `remote` – путь до файла на сервере
- `local` – путь куда сохранить файл
- `preserve_mode` – сохранить права

```
carnival.cmd.transfer.put(local: str, remote: str, preserve_mode: bool = True)
    → <sphinx.ext.autodoc.importer._MockObject object at
    0x7f7f9b7b3850>
```

Закачать файл на сервер <<http://docs.fabfile.org/en/2.5/api/transfer.html#fabric.transfer.Transfer.put>>

#### Параметры

- `local` – путь до локального файла
- `remote` – путь куда сохранить на сервере
- `preserve_mode` – сохранить права

```
carnival.cmd.transfer.put_template(template_path: str, remote: str, **context) →
    <sphinx.ext.autodoc.importer._MockObject object at
    0x7f7f9b7b3850>
```

Отрендерить файл с помощью jinja-шаблонов и закатать на сервер См раздел templates.

<<http://docs.fabfile.org/en/2.5/api/transfer.html#fabric.transfer.Transfer.put>>

#### Параметры

- `template_path` – путь до локального файла jinja
- `remote` – путь куда сохранить на сервере
- `context` – контекст для рендеринга jinja2

```
carnival.cmd.transfer.rsync(source: str, target: str, exclude: Iterable[str] = (), delete: bool = False,
    strict_host_keys: bool = True, rsync_opts: str = '-progress -pthrvz',
    ssh_opts: str = '') → <sphinx.ext.autodoc.importer._MockObject
    object at 0x7f7f9b7b3850>
```

<<https://fabric-patchwork.readthedocs.io/en/latest/api/transfers.html#patchwork.transfers.rsync>>



---

Indices and tables

---

- `genindex`
- `modindex`
- `search`





## C

- `carnival.cli`, [9](#)
- `carnival.cmd`, [13](#)
- `carnival.cmd.appt`, [13](#)
- `carnival.cmd.cli`, [14](#)
- `carnival.cmd.fs`, [14](#)
- `carnival.cmd.system`, [15](#)
- `carnival.cmd.systemd`, [16](#)
- `carnival.cmd.transfer`, [17](#)
- `carnival.host`, [1](#)
- `carnival.internal_tasks`, [8](#)
- `carnival.templates`, [11](#)



### СИМВОЛЫ

`__init__()` (метод *carnival.Step*), 5  
`__init__()` (метод *carnival.host.Host*), 2  
`__init__()` (метод *carnival.host.LocalHost*), 3  
`__init__()` (метод *carnival.host.SSHHost*), 3

### А

`AnyConnection` (в модуле *carnival.host*), 1

### С

`call_task()` (метод *carnival.Task*), 7  
`carnival.cli` (модуль), 9  
`carnival.cmd` (модуль), 13  
`carnival.cmd.apt` (модуль), 13  
`carnival.cmd.cli` (модуль), 14  
`carnival.cmd.fs` (модуль), 14  
`carnival.cmd.system` (модуль), 15  
`carnival.cmd.systemd` (модуль), 16  
`carnival.cmd.transfer` (модуль), 17  
`carnival.host` (модуль), 1  
`carnival.internal_tasks` (модуль), 8  
`carnival.templates` (модуль), 11

### Д

`daemon_reload()` (в модуле *carnival.cmd.systemd*), 16  
`disable()` (в модуле *carnival.cmd.systemd*), 16

### Е

`enable()` (в модуле *carnival.cmd.systemd*), 16  
`ensure_dir_exists()` (в модуле *carnival.cmd.fs*), 14  
`except_hook()` (в модуле *carnival.cli*), 9

### Ф

`force_install()` (в модуле *carnival.cmd.apt*), 13

### Г

`get()` (в модуле *carnival.cmd.transfer*), 17

`get_current_user_id()` (в модуле *carnival.cmd.system*), 15  
`get_current_user_name()` (в модуле *carnival.cmd.system*), 15  
`get_installed_version()` (в модуле *carnival.cmd.apt*), 13  
`get_pkg_versions()` (в модуле *carnival.cmd.apt*), 13

### Н

`help` (атрибут *carnival.internal\_tasks.Help*), 8  
`Help` (класс в *carnival.internal\_tasks*), 8  
`host` (атрибут *carnival.host.LocalHost*), 2, 3  
`Host` (класс в *carnival.host*), 1, 2

### И

`install()` (в модуле *carnival.cmd.apt*), 13  
`install_multiple()` (в модуле *carnival.cmd.apt*), 13  
`is_completion_script()` (в модуле *carnival.cli*), 9  
`is_connection_local()` (метод *carnival.host.SSHHost*), 2, 3  
`is_current_user_root()` (в модуле *carnival.cmd.system*), 15  
`is_dir_exists()` (в модуле *carnival.cmd.fs*), 14  
`is_file_contains()` (в модуле *carnival.cmd.fs*), 14  
`is_file_exists()` (в модуле *carnival.cmd.fs*), 15  
`is_pkg_installed()` (в модуле *carnival.cmd.apt*), 14

### Л

`LOCAL_ADDRS` (атрибут *carnival.host*), 2  
`LocalHost` (класс в *carnival.host*), 2, 3

### М

`main()` (в модуле *carnival.cli*), 9  
`makedirs()` (в модуле *carnival.cmd.fs*), 15  
`module_name` (атрибут *carnival.internal\_tasks.Help*), 8

**P**

`pty()` (в модуле *carnival.cmd.cli*), 14  
`put()` (в модуле *carnival.cmd.transfer*), 17  
`put_template()` (в модуле *carnival.cmd.transfer*),  
17

**R**

`remove()` (в модуле *carnival.cmd.apd*), 14  
`render()` (в модуле *carnival.templates*), 11  
`restart()` (в модуле *carnival.cmd.systemd*), 16  
`rsync()` (в модуле *carnival.cmd.transfer*), 17  
`run()` (метод *carnival.internal\_tasks.Help*), 8  
`run()` (метод *carnival.Step*), 5  
`run()` (метод *carnival.Task*), 7  
`run()` (в модуле *carnival.cmd.cli*), 14

**S**

`set_password()` (в модуле *carnival.cmd.system*), 15  
`SimpleTask` (класс в *carnival*), 7  
`ssh_authorized_keys_add()` (в модуле  
*carnival.cmd.system*), 15  
`ssh_authorized_keys_ensure()` (в модуле  
*carnival.cmd.system*), 15  
`ssh_authorized_keys_list()` (в модуле  
*carnival.cmd.system*), 16  
`ssh_copy_id()` (в модуле *carnival.cmd.system*), 16  
`SSHHost` (класс в *carnival.host*), 2, 3  
`start()` (в модуле *carnival.cmd.systemd*), 16  
`Step` (класс в *carnival*), 5  
`step()` (метод *carnival.Task*), 7  
`stop()` (в модуле *carnival.cmd.systemd*), 16

**T**

`Task` (класс в *carnival*), 7  
`TaskResult` (класс в *carnival.task*), 8