?????????? ? ??????

??????

> **Модуль** — это блок, описывающий настройку одной из частей системы, используя опции как способ взаимодействия с другими блоками, настройкой системы в целом или отдельными её частями.

Объявление этих опций является его основным отличием от других блоков с настройками. Модули позволяют разбивать настройку системы на части, делая ее более структурированной, повторно используемой и масштабируемой.

??????? ??????

1. **Абстракция.** Модули скрывают детали реализации, предоставляя стандартизированный способ взаимодействия для настройки компонентов системы. В этом смысле он больше похож на описание класса в языке программирования.
2. **Повторное использование.** Один и тот же модуль может использоваться в разных частях системы или в разных конфигурациях для однотипных настроек.
3. **Использование других модулей**. Модули могут импортировать другие модули, позволяя создавать иерархическую структуру конфигурации.

????????? ??????

1. Импорт других модулей(**imports**).
2. Объявление опций(**options**).
3. Определение опций и описание настроек модуля(**config**).

```
{
  imports = [
    # Пути к другим модулям.
    # Модули могут импортировать другие модули, позволяя
    # создавать иерархическую структуру настроек.
  ];

  options = {
```

```
    # Объявление опций.
    # Объявляет какие настройки пользователь этого модуля может устанавливать.
    # Обычно это включает общий пункт "enable" изначально установленный в ложное значение.
  };

  config = {
    # Определение опций.
    # Определяет какие другие настройки, службы и ресурсы должны быть задействованы.
    # Обычно это зависит от того выбрал ли пользователь этого модуля
    # пункт "enable" используя объявление выше.
    # Опции для модулей импортированных в секции "imports" могут быть установлены здесь.
  };
}
```

Каждый модуль может объявлять новые опции, которые являются настройками для других частей системы. Например, модуль для настройки веб-сервера может иметь опции порта и директории сайта.

Все модули получают доступ к переменной `config`, которая содержит текущие определения опций из других модулей, что позволяет модулям взаимодействовать друг с другом и использовать ранее определенные настройки.

Поскольку модули описывают желаемое состояние системы, а не шаги по его достижению, NixOS является декларативной системой, где настройка системы осуществляется средствами модулей.

Пример:
Вот как может выглядеть упрощенный модуль для гипотетической службы `my-service`:

```
# ./my-module.nix
{ config, lib, pkgs, ... }: # Это функция, принимающая аргументы

{
  # 1. Объявляем опции, которые можно использовать в configuration.nix
  options = {
    services.my-service = {
      enable = lib.mkEnableOption "My cool service"; # Создает опцию enable с описанием
      port = lib.mkOption {
        type = lib.types.port;
        default = 8080;
        description = "Port for my-service";
      };
```

```
    };
  };

  # 2. Определяем, что делать, если служба включена
  config = lib.mkIf config.services.my-service.enable {
    # Это "вклад" модуля в общую конфигурацию системы
    environment.systemPackages = [ pkgs.my-service-pkg ]; # Добавляем пакет в окружение
    systemd.services.my-service = { # Создаем systemd юнит
      description = "My Service";
      after = [ "network.target" ];
      wantedBy = [ "multi-user.target" ];
      serviceConfig = {
        ExecStart = "${pkgs.my-service-pkg}/bin/my-service --port ${toString
config.services.my-service.port}";
      };
    };
  };
}
```

Теперь в `configuration.nix` мы можем **импортировать** этот модуль и использовать его опции:

```
# /etc/nixos/configuration.nix
{ config, pkgs, ... }:

{
  imports = [ ./my-module.nix ]; # Импортируем модуль

  # Используем опции, объявленные в модуле
  services.my-service = {
    enable = true;
    port = 9000;
  };

  # ... остальная конфигурация системы
}
```

??????? ????????????? ???????

Модули могут использоваться для:

1. **Управления службами**. Модуль может описывать настройку и запуск службы, например, веб-сервера nginx, включая его зависимости.
2. **Настройки пользователя.** NixOS home-manager использует модули для управления настройками пользователя и содержимым его домашнего каталога, например, настройками оболочки или приложения.
3. **Описание пакетов.** Модули могут описывать сборку и установку пакетов, а также управлять их зависимостями.

?????????? ???????

1. **Воспроизводимость.** Модули упрощают создание воспроизводимых настроек, которые можно легко применить на разных машинах.
2. **Управление.** Разбиение настроек на модули делает ее более управляемой и понятной, даже для сложных систем.
3. **Повторное использование.** Модули могут быть повторно использованы в разных настройках, что сокращает дублирование кода.

????????? ??????

> **Вложенные модули** — это тип, позволяющий определять вложенные модули с собственным набором опций. Он используется для структурирования сложных настроек, группируя связанные опции в отдельный блок.

При использовании `lib.types.submodule` вы создаёте новую опцию, значением которого будет набор атрибутов, соответствующих другому, вложенному модулю. Этот вложенный модуль имеет свой собственный набор опций, как и обычный модуль NixOS.

Пример:
Допустим, вы создаёте модуль для веб-сервиса, и хотите, чтобы опции для базы данных были сгруппированы.

1. **Объявление типа `submodule`:**
   Вы определяете опцию `database` с типом `submodule`. Внутри `submodule` вы задаёте собственные опции, например, `host`, `port` и `user`.

```
# module.nix
{ config, lib, ... }:

let
  types = lib.types;
in
```

```
{
  options.database = lib.mkOption {
    type = types.submodule {
      options = {
        host = lib.mkOption {
          type = types.str;
          default = "localhost";
        };
        port = lib.mkOption {
          type = types.int;
          default = 5432;
        };
        user = lib.mkOption {
          type = types.str;
          default = "admin";
        };
      };
    };
    description = "Настройки для базы данных.";
  };
}
```

2. **Использование в настройке:**
   Затем в вашей настройке вы можете определить значения для этих вложенных опций.

```
# configuration.nix
{ ... }:

{
  imports = [
    ./module.nix
  ];

  database = {
    host = "db.example.com";
    port = 5433;
    user = "web_user";
  };
}
```

В результате вы получите настройки, где `config.database.host` будет `"db.example.com"`, `config.database.port` будет `5433` и `config.database.user` будет `"web_user"`.

??????????? ????????? ???????

- **Модульность:** Позволяет разбивать сложные настройки на логические, повторно используемые блоки.
- **Структурирование:** Повышает читаемость и удобство управления настройками.
- **Типобезопасность:** Каждая опция внутри `submodule` имеет свой тип и проверяется, что предотвращает ошибки настройки.
- **Совместимость с другими типами:** Часто используется в сочетании с `lib.types.attrsOf` или `lib.types.listOf` для создания списков или наборов подмодулей. Например, можно создать `listOf submodules` для определения нескольких веб-сервисов с похожими настройками.

????????????

> **Конфигурация** – это блок, описывающий настройку системы в целом или отдельной её части.

В NixOS файл конфигурации `/etc/nixos/configuration.nix` предназначен для описания всех настроек системы. Он может включать другие файлы конфигурации, описывающие только часть настроек общей системы, а также модули, изменять настройки которых можно через их опции. **Конфигурация** описывает **заданное**, **статическое состояние** системы, в то время как **модули могут изменять свое поведение в зависимости от значения, передаваемое их опциям**. Например, вы можете указать, какие сайты разместить на nginx, используя для этого модуль.

???????? ????? ??????? ? ????????????

Разница между модулем и конфигурацией в NixOS заключается в их функциях и способе использования. Модуль представляет собой часть кода Nix, которую можно настроить для создания конфигурации . Конфигурация, в свою очередь, является результатом работы модулей и определяет параметры всей системы .

Основные моменты, поясняющие разницу:

1. **Модули**:

- **Функция**: Модуль — это функция, которая принимает набор атрибутов и возвращает другой набор атрибутов .
- **Назначение**: Модули позволяют организовывать конфигурацию системы, абстрагируя сложные настройки и предоставляя возможность повторного использования кода .
- **Пример**: Модули могут использоваться для настройки служб, параметров загрузки и других системных компонентов .

2. **Конфигурация**:

- **Функция**: Конфигурация — это набор настроек, определяющих поведение системы.
- **Назначение**: Конфигурация создается на основе модулей и содержит все параметры, необходимые для работы системы.
- **Пример**: Конфигурационный файл `configuration.nix` в NixOS определяет всю конфигурацию системы, используя различные модули .

В заключение, модули в NixOS используются для организации и модульности конфигурации, в то время как конфигурация является конечным результатом работы этих модулей, определяющим настройки системы.

??????????

- **Опция** — необязательная возможность, меняющая поведение функции. Опция может иметь один или набор параметров.

?????? ????????????? ??????????

1. NixOS modules
2. Understanding NixOS Modules and Declaring Options

Revision #19
Created 2025-10-27 13:37:41 UTC by Антон Сергеевич Абраменко
Updated 2025-11-25 04:49:04 UTC by Антон Сергеевич Абраменко