Завдання : Неблокуючи запити, звичайний сокет, C++. У якості носіїв компонентів обчислення розглядаються процеси.

Ключові компоненти

Класи для обчислень (стратегії):

- **`CalculationStrategy`**:
  - Абстрактний базовий клас для визначення стратегії обчислень.
  - Має єдиний віртуальний метод `calculate(int x)`, який реалізується у спадкових класах.
- **Конкретні реалізації**:
  - **`FunctionG`**: обчислює `x * x * 1.5`.
  - **`FunctionH`**: обчислює квадратний корінь `sqrt(x)`.
  - **`FunctionF`**: обчислює куб `x * x * x`.

Обробка сигналів:

- **`signalHandler(int signal)`**:
  - Обробляє сигнал `SIGINT` (Ctrl+C), завершуючи програму.
- **`sigchldHandler(int signal)`**:
  - Чекає завершення дочірніх процесів, щоб уникнути "зависання" зомбі-процесів.

Клас `Manager` (Менеджер компонентів):

- Відповідає за управління групами задач, їх додавання, виконання та перегляд результатів.
- Основні методи:
  - **`createGroup(const std::string& name)`**: створює нову групу задач.
  - **`addComponent(const std::string& name, std::shared_ptr<CalculationStrategy>, int arg)`**: додає компонент до поточної групи.
  - **`run()`**: запускає всі компоненти у поточній групі у вигляді окремих процесів.
  - **`status()`**: відображає статус усіх компонентів.
  - **`summary()`**: відображає результати виконання.
  - **`clear()`**: очищає групу і завершує роботу активних процесів.

Клас `Command` (Команди):

- Базовий клас для створення команд.
- Конкретні реалізації:
  - **`CreateGroupCommand`**: створює нову групу.
  - **`AddComponentCommand`**: додає компонент до групи.
  - **`RunCommand`**: запускає всі компоненти.

Клас `ComponentFactory`:

- Створює компоненти для виконання обчислень на основі заданого типу (`"g"`, `"h"`, `"f"`).

Команди інтерфейсу

1. **`help`**:
   - Виводить список доступних команд.
2. **`group <name>`**:
   - Створює нову групу задач із назвою `<name>`.
3. **`new <component> <arg>`**:
   - Додає новий компонент до групи.
   - `<component>`: тип компонента (`"g"`, `"h"`, `"f"`).
   - `<arg>`: аргумент для обчислення.
4. **`run`**:
   - Запускає виконання всіх компонентів у поточній групі.
5. **`status`**:
   - Показує статус усіх активних компонентів (ім'я і PID процесу).
6. **`summary`**:
   - Виводить результати виконання кожного компонента.
7. **`clear`**:
   - Завершує роботу всіх компонентів і очищує поточну групу.
8. **`exit`**:
   - Завершує роботу програми.

Приклад використання

1. Створити групу:
   ```
   > group MyGroup
   ```
2. Додати компоненти:
   ```
   > new g 10
   ```
   > new h 25
3. Запустити обчислення:
   ```
   > run
   ```
4. Перевірити статус:
   ```
   > status
   ```

5. Отримати результати:
   ```
   > summary
   ```

6. Очистити групу:
   ```
   > clear
   ```

7. Вийти:
   ```
   > exit
   ```