Отлично! Давайте кратко разберем теорию, отталкиваясь от вашего кода и вывода strace.

**1. Процессы в операционной системе**

Процесс — это экземпляр программы, который выполняется в операционной системе. Каждый процесс имеет свой уникальный идентификатор (PID) и изолированное пространство памяти.

* **fork()** — системный вызов для создания нового процесса. Он создаёт копию родительского процесса. В родительском процессе возвращается PID дочернего процесса, а в дочернем процессе возвращается 0.
* **execve()** — замещает текущий процесс другим. В вашем выводе strace это видно через execve("./parent", ...), когда родительский процесс запускает программу.
* **wait() / wait4()** — используется родительским процессом для ожидания завершения дочернего процесса. В вашем выводе, например, это выражено через:
* wait4(-1, NULL, 0, NULL) = 1833

**2. Межпроцессное взаимодействие (IPC)**

Межпроцессное взаимодействие (IPC, Inter-Process Communication) — это механизмы, которые позволяют процессам обмениваться данными и синхронизировать свои действия.

* **Каналы (pipes)** — одно из средств IPC. В вашем случае, в выводе strace, видно создание канала:
* pipe2([3, 4], 0) = 0

Канал состоит из двух файловых дескрипторов: один для чтения (в вашем случае 3), другой для записи (4). Родительский процесс записывает данные в канал, а дочерний процесс читает их.

**3. Системные вызовы, используемые в вашем коде:**

1. **pipe2()** — создаёт два файловых дескриптора для канала, которые могут использоваться для передачи данных между процессами. В вашем коде родительский процесс передает данные через канал дочернему процессу, используя write() и read().
2. **fork() / clone()** — используется для создания нового процесса. В вашем случае это видно в выводе wait4(), где родительский процесс ожидает завершения дочернего процесса с PID 1833. clone() — это один из вариантов вызова, который также может быть использован для создания новых процессов в Linux.
3. **execve()** — используется для запуска программы внутри процесса. В вашем случае это можно увидеть в том, как родительский процесс запускает программу:
4. execve("./parent", ["./parent"], 0x7ffdc1c8b330 /\* 28 vars \*/) = 0

**4. Типичный процесс взаимодействия:**

* Родительский процесс создаёт канал (pipe2()).
* Родительский процесс запрашивает у пользователя ввод чисел и передает их через канал.
* Дочерний процесс читает данные из канала, обрабатывает их (в вашем случае — считает сумму чисел).
* Дочерний процесс записывает результат в файл.
* Родительский процесс ожидает завершения дочернего процесса с помощью wait4().

**5. Что можно ожидать на защите?**

1. Вопросы могут касаться **создания процессов**: например, как работает fork(), что происходит в родительском и дочернем процессах, когда вызывается fork().
2. **Межпроцессное взаимодействие**: как происходит передача данных между процессами (через каналы), как можно использовать другие методы IPC (например, мьютексы, семафоры).
3. Вопросы по **системным вызовам**: что делает execve(), зачем он нужен и как используется в контексте вашего кода.
4. **Что происходит в выводе strace**: интерпретация того, как создаются процессы и каналы, а также как выглядит взаимодействие процессов.

**Пример возможного вопроса и ответа:**

**Вопрос:** Как работает межпроцессное взаимодействие в вашей программе с использованием канала?

**Ответ:** В моей программе родительский процесс создает канал с помощью pipe2(), который создаёт два файловых дескриптора. Родитель передает данные в канал с помощью системного вызова write(), а дочерний процесс читает эти данные через read(). После обработки данных (суммирования чисел) дочерний процесс записывает результат в файл, и родительский процесс ожидает завершения дочернего процесса с помощью вызова wait4().

Надеюсь, это поможет вам на устной защите! Если нужно больше теории или примеров, дайте знать.