Задание

По пути */home/<your_username>/src/simple_example_custom* расположить код другой программы. Это может быть либо программа на C, либо программа на каком либо другом языке программирования, желательно компилируемом. В последнем случае там же разместить скомпилированный бинарник. Требуется написать свою версию программы так, чтобы она содержала хотя бы на один системный вызов больше предыдущей. Новые вызовы также описать. Если Вы решили использовать интерпретируемый язык программирования, без компиляции в бинарник, то выполните трассировку при помощи `strace -F -c`.

Решение

Код программы */home/usero1/src/simple_example_custom/**fork.c***

```
#include <stdio.h>
#include <unistd.h>

int main(void) {
    fork();
    puts("Hello, Fintech World!");
    return 0;
}
```

Появляется один новый вызов:

- clone (системный вызов) — создаёт новый процесс, и выполнение в потомке продолжается от места данного вызова.

  Аргумент *child_stack* задаёт положение стека, используемого процессом-потомком: вызывающий процесс устанавливает пространство памяти для стека процесса-потомка и передать указатель на это пространство в вызове *clone()*. Стеки растут вниз для всех процессоров, поэтому *child_stack* обычно указывает на наиболее высокий адрес в пространстве памяти, которое устанавливается для стека процесса-потомка.

  Аргумент *flags* состоит из одного или более битовых флагов, задающих порядок процессов и то, что разделяется между вызывающим процессом и процессом-потомком.

  Младший байт *flags* содержит номер сигнала, который посылается родителю, когда потомок умирает. Если никакой сигнал не задан, то родительский процесс не извещается сигналом, когда потомок завершается.

  Начиная с Linux 2.5.49 у системного вызова может быть ещё два агрумента:

  - *parent_tidptr* — указывает расположение в памяти (отца и сына) значения child thread ID (если установлен нужный флаг)
  - *child_tidptr* — указывает расположение в памяти (сына) значения child thread ID (если установлен другой нужный флаг)

  В случае успеха *clone()*, в вызывающий тред возвращается PID процесса-потомка. В случае ошибки, в контекст вызываюего процесса возвращается -1, процесс потомок не будет создан и значение *errno* устанавливается

соответствующим образом.

```
int clone(void *child_stack, int flags)
```