Задача 5. Плагины

Источник:	основная I
Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	разумное
Ограничение по памяти:	разумное

В продолжение предыдущей задачи, предлагается реализовать минимальный строковый “ассемблер”, команды которого загружаются из плагинов — динамических библиотек.

У процессора есть состояние `State` — это массив из 256 строковых “регистров”.

```
typedef struct State {
    char *regs[256];
} State;
```

Каждый регистр либо нулевой (т.е. указатель равен 0), либо указывает на C-шную строку, расположенную на куче.

На стандартный поток ввода поступают команды, их все нужно выполнить в порядке их поступления. Каждая строка ввода — это одна команда, полная длина команды не превышает 1000 символов. Команда состоит из слов, разделённых проблеми (внутри слов пробелов нет). Первое слово в команде — это имя команды, а остальные слова (от нуля до трёх штук включительно) — это аргументы.

Код реализации команд разбросан по динамическим библиотекам, “плагинам”. Имя команды записывается в формате: сначала имя плагина, потом двоеточие, потом имя функции. Как имя плагина, так и имя функции — это строки длиной не более 30 символов, состоящие только из латинских букв и цифр. Имя плагина может быть опущено: в таком случае имя команды состоит лишь из имени функции, а имя плагина считается автоматически равным “`core`”.

Чтобы выполнить команду, ваша программа должна загрузить динамическую библиотеку из файла, имя которого совпадает с именем плагина (расширение `.so`). После этого нужно найти в этой библиотеке функцию, имя которой составляется как: имя функции, подчерк, количество аргументов. Далее нужно запустить эту функцию, передав ей сначала указатель на `State` (глобальное состояние програмы), а затем указатели на аргументы команды.

Рассмотрим примеры:

- `core:echo hello world`

  Здесь имя команды `core:echo`, имя плагина `core`, имя функции `echo`, два аргумента `hello` и `world`. Нужно выполнить функцию с именем `echo_2` из динамической библиотеки `core.so`:

  ```
  //arg0 = 'hello', arg1 = 'world'
  echo_2(&state, arg0, arg1);
  ```

- `echo a b c`

  Здесь имя команды `echo`, значит имя плагина по умолчанию `core`. Нужно вызвать из `core.so` функцию `echo_3`, передав все три строки как аргументы:

```
//arg0 = 'a', arg1 = 'b', arg2 = 'c'
echo_3(&state, arg0, arg1, arg2);
```

- `string:tokenize mama_mila__ramu_`
  `printregs`

  Здесь две команды. В первой нужно выполнить функцию `tokenize_1` из `string.so`, передав ей указанную строку как аргумент. Во второй нужно выполнить `printregs_0` из `core.so`.

Наверное вы уже заметили, что библиотека `core.so` из первой задачи отлично работает в качестве плагина к этой задаче.

Кроме того, в этой задаче требуется реализовать второй плагин `string.so` со следующими функциями:

```
//loads string A from I-th register ([idx0] contains its index)
//loads string B from J-th register ([idx1] contains its index)
//then stores concatenation of A and B into I-th register
//BEWARE: [idx0] and [idx1] are allowed to be equal indices
void concat_2(State *state, char *idx0, char *idx1);
//extracts sequence of tokens/words separated by underscore character from string [arg]
//puts K -- number of tokens into 0-th register
//puts the tokens into 1-th, 2-th, ..., K-th registers (in correct order)
void tokenize_1(State *state, char *arg);
```

Кроме того, от вас требуется собственно реализовать строковый ассемблер — исполняемый файл, работающий по описанным правилам. Он должен обрабатывать ошибку при загрузке функции:

1. Если для выполняемой команды нет файла динамической библиотеки с заданным именем, надо выдать сообщение "`Missing plugin имяплагина`". Здесь имя плагина берётся без расширения `.so`.
2. Если файл плагина есть, но в нём нет нужной функции, нужно выдать сообщение "`Missing function имяфункции in plugin имяплагина`". Здесь имя функции должно включать подчёркивание и количество аргументов.

Других проблем при тестировании **не** будет.

При тестировании будут использоваться в том числе плагины жюри. Все имеющиеся в наличии плагины будут находиться в рабочей директории, там же будет запускаться ваша программа.

В ходе запуска одного теста будет использовано не более 35 различных плагинов. Количество выполненных команд в каждом тесте не превышает 20 000.

Требуется отправить в систему тестирования набор исходных файлов, а также два скрипта сборки. Скрипт `string.bat.txt` должен собирать динамическую библиотеку `string.so` с указанными выше двумя функциями. Скрипт `myasm.bat.txt` должен собирать исполняемый файл `myasm` строкового ассемблера.

Скрипты должны иметь расширение `.sh.txt`, библиотеки — `.so`, а исполняемый файл должен быть без расширения.

Пример

В примере полагается, что есть только два плагина: `core.so` из задачи 1 и `string.so`, описанный выше.

stdin

`string:tokenize mama_mila__ramu_`
`printregs`
`concat 1 2`
`string:concat 1 2`
`clear 3`
`printregs`
`string:concat 2 1`
`print 2`
`omg:sos 666`

stdout

`0 = 3`
`1 = mama`
`2 = mila`
`3 = ramu`
`Missing function concat_2 in plugin core`
`0 = 3`
`1 = mamamila`
`2 = mila`
`milamamamila`
`Missing plugin omg`

Комментарий

Обратите внимание, что поведение программы при запуске зависит от многих факторов. Например, собран Debug, Release или ручная сборка из командной строки, запущена программа с отладкой Visual Studio или напрямую, как подаются данные на вход и вытягиваются на выходе.

Если вы получаете Wrong Answer на примере, но локально у вас пример работает, рекомендуется сделать так:

1. Соберите программу и библиотеку, запустив свои .bat-файлы.
2. Создайте файл `input.txt` и скопируйте в него входные данные из примера.
3. Запустите в командной строке: `myasm <input.txt >output.txt`
4. Проверьте полученный вывод в файле.

Знак “меньше” в консоли перенаправляет содержимое заданного файла в stdin. Аналогично работает знак “больше” с stdout.