Куда идём мы с Пятачком?

Сказка о том, как можно заблудиться в потоках

Открывай, сова, медведь пришёл!

```
$ echo “Some message”
Some message

# Мы идём в консоль
```

А не пойти ли нам в гости?

```
$ echo “Some message” >some_file
$ cat some_file
Some message

# Мы идём в файл
```

А если хочется зайти в оба места?

А если хочется зайти в оба места?

```
$ echo “Some message” | tee -a some_file
Some message
$ cat some_file
Some message

# Мы идём и в консоль, и в файл
```

Никто не спросил себя: Где Пятачок?

Никто не спросил себя: Где Пятачок?

```
$ echo “Some error” >&2 | tee -a some_file
Some error
$ cat some_file

# Оп-па, ничего не пришло
```

Никто не спросил себя: Где Пятачок?

```
$ echo “Some error” 2>&1 >&2 | tee -a some_file
Some error
$ cat some_file
Some error

# Ну вот, Пятачок тоже с нами
```

Чем дальше в лес …

Чем дальше в лес …

```
3>&1 2>&1 1>&2 3>&- | tee -a some_file

# Подобная конструкция в конце команды поможет
# нам отправить stderr и в файл, и на экран, но
# stdout на этот раз будет только на экране
```

Чем дальше в лес ...

```
> >(tee -a some_file) 2> >(tee -a some_errors)*

# А такое даже позволит нам писать и в консоль,
# и в разные файлы для разных потоков
```

Чем дальше в лес ...

```
> >(tee -a some_file) 2> >(tee -a some_errors)*

# А такое даже позволит нам писать и в консоль
# и в разные файлы для разных потоков

* Внимание: Так называемый башизм
```

И вот новая задача от кролика:
теперь мы должны создать скрипт

Что делать, шеф?

Как перенаправить все команды?

Как перенаправить все команды?

Решение 1.

Писать перенаправление для каждой команды

Как перенаправить все команды?

Решение 2.

Вызывать сам скрипт, задав нужные направления именно для него

Как перенаправить все команды?

Решение 3.

Использовать специальный вызов exec внутри скрипта

```
exec >some_file 2>/dev/null
```

Как перенаправить все команды?

Решение 4.
Воспользоваться библиотекой sh.adow_streamer

```
# Внезапно, это самореклама :)
```

https://github.com/gitlava/sh.adow_streamer

+ Перенаправление и мультиплицирование стандартных потоков;

+ Открытие новых потоков для нужд отладки с лёгкой возможностью их отключения;

+ Возможность задания динамичного префикса при выводе сообщений.

lava@surtr:~/Dropbox/github/sh.adow_streamer

```
$ cat ./test.sh
#!/bin/sh

. /usr/lib/sh.adow/sh.adow_streamer

wrp_level 1 -p '[INFO] `date -Is`: '  - ./stdout
wrp_level 2 -p '[ERROR] `date -Is`: ' - ./stderr
wrp_level 4 -p '[DEBUG] `date -Is`: ' ./level-4
wrp_level 6 -- -p '[DEBUG] `date -Is`: ' ./level-4

echo Normal message
echo Error happens >&2
echo Level-4 message >&4
echo Level-6 message >&6

sleep 1

$
```

lava@surtr:~/Dropbox/github/sh.adow_streamer

```
$ ./test.sh
[INFO] 2012-06-08T18:49:11+0300: Normal message
[ERROR] 2012-06-08T18:49:11+0300: Error happens
$ cat ./stdout
[INFO] 2012-06-08T18:49:11+0300: Normal message
$ cat ./stderr
[ERROR] 2012-06-08T18:49:11+0300: Error happens
$ cat ./level-4
[DEBUG] 2012-06-08T18:49:11+0300: Level-4 message
$
```

Известные проблемы

- **Процессы перенаправления потоков совершенно независимы и, как следствие:**
  - **а. конкурентны;**
  - **б. не отдают вывод дальше по конвейеру.**

И они посидели ещё немножечко. А потом ещё немножечко… И ещё немножечко… Пока, увы, совсем ничего не осталось!

Конец

Контакты

github: https://github.com/gitlava
xmpp: lava@midgard.by