04. Эффективная работа с командной строкой

Раздел 1

Терминалы

Назначение терминалов

- **Терминал** - устройство, предназначенные для ввода и вывода данных и организации таким образом человеко-машинного интерфейса.
- **алфавитно-цифровые** терминалы - позволяют реализовать TUI и CLI
- **графические** терминалы - позволяют реализовать GUI, а так же TUI и CLI за счет приложений-эмуляторов

- **Ввод и вывод данных** - терминал состоит из устройств вывода - дисплей (ранее м.б. принтер) и ввода - клавиатура, мышь и т.п.
- **Консоль** - термин, который часто используется вместо термина терминал либо как синоним для алфавитно-цифрового терминала. Сочетание устройств ввода-вывода (видеоадаптер, монитор и клавиатура), используемое для организации взаимодействия с пользователем.

Организация командного интерфейса

- Двусторонний попеременный диалог между пользователем и операционной системой (процессами) - процесс ввода команд пользователем посредством клавиатуры и получения результата их выполнения на дисплее терминала.

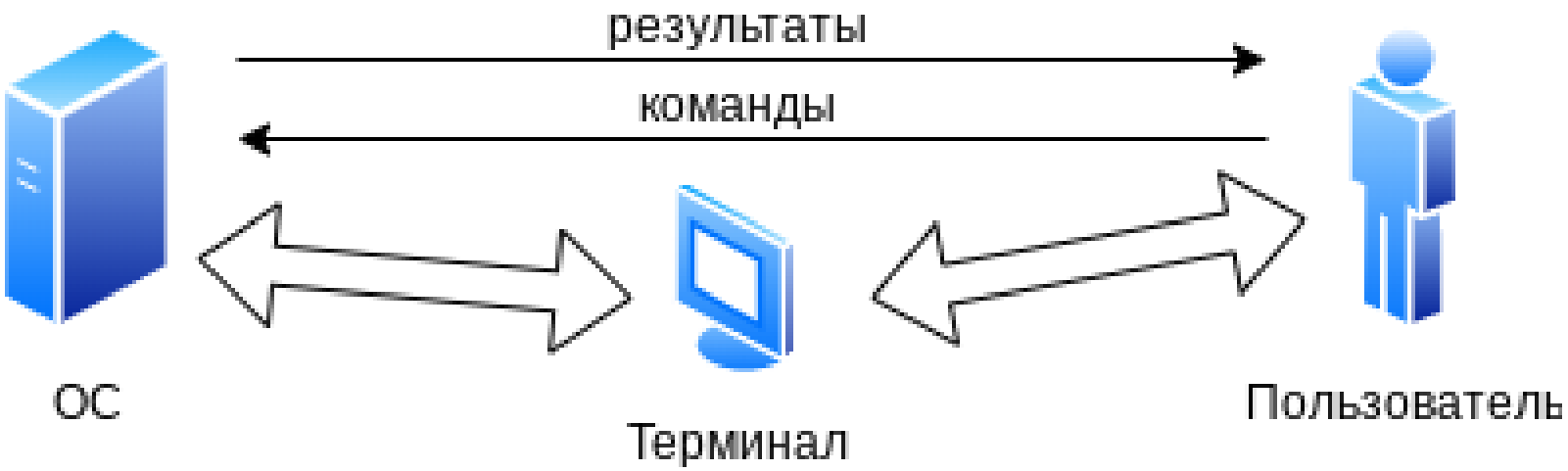

Рисунок 1: Двусторонний попеременный диалог

- Соответствие процессов и терминалов - процессы, предусматривающие взаимодействие с пользователем посредством TUI и CLI имеют привязку к терминалу, т.н. управляющий терминал, посредством которого и взаимодействуют с пользователем.

```
$ ps
    PID TTY          TIME CMD
   7309 pts/1    00:00:00 bash
  10521 pts/1    00:00:00 ps
```

- Список процессов - командный интерпретатор и команда ps, сопоставленные текущему терминалу - pts/1

Драйвер терминала (tty)

- Программное обеспечение, обеспечивающее передачу команд от устройств ввода пользовательскому процессу, и результатов выполнения команд от пользовательского процесса на устройства вывода.
- **Управление драйвером терминала** производится путем обработки специальных управляющих символов и ESC-последовательностей. Драйвер, например, может обеспечивать возможности произвольного позиционирования курсора или изменять цвет отображения символов.

Виды терминалов

- Виды терминалов с точки зрения реализации
  - аппаратные
  - виртуальные
  - псевдотерминалы

Аппаратные или физические терминалы

Специальные аппаратные устройства, включающие дисплей (как правило алфавитно-цифровой), клавиатуру, последовательный порт для взаимодействия с ЭВМ и специальное программное обеспечение для обработки управляющих последовательностей символов. В настоящий момент большая редкость.

Виртуальные терминалы

- Терминалы, эмулируемые специальным драйвером (драйвером виртуальных терминалов) для организации взаимодействия с пользователем через сочетание консольных устройств.

`Ctrl+Alt+F1 (2-6)`

- Переключение на виртуальные терминалы. При установки компонент графического интерфейса, обычно первый виртуальный терминал используется как графический, остальные со 2 по 6ой - как алфавитно-цифровые.

Псевдотерминалы

Терминалы, эмулируемые специальной программой-посредником (программным эмулятором терминала) для организации взаимодействия с пользователем

- Графические эмуляторы терминала (xterm, mate-terminal, konsole, и т.д.) - приложения, эмулирующие алфавитно-цифровой терминал поверх графического для возможности работы с интерфейсом командной строки в графической среде.
- Эмуляторы терминала при удаленном терминальном доступе - аналогично при подключении пользователя посредством протокола удаленного терминального доступа (например telnet, ssh), соответствующее приложение клиент выполняет эмуляцию терминала.

Режимы ввода-вывода для алфавитно-цифрового терминала

- **Канонический режим** (canonical или cooked) - ввод данных происходит построчно, завершается символом перевода строки. До ввода завершающего символа есть возможность редактировать строку. Канонический режим используется для организации командного интерфейса и простейших TUI.
- **Неканонический режим** (non-canonical или raw) - данные не группируются в строки, а передаются группами символов. Используется полноэкранными утилитами (vim, less, а так же построенных на библиотеке ncurses - mc и т.п.). В неканоническом режиме терминал рассматривается как матрица символов, которыми можно произвольно управлять.

Управляющие символы

- Управление драйвером терминала при **вводе** - пользователь, взаимодействуя с терминалом, вводит специальные символы, которые служат командами для драйвера терминала. Тем самым реализуется управление курсором в командной строке, либо посылка сигналов процессу, обрабатывающему ввод с терминала.

Символ	Объект	Действие
Ctrl-c	Процесс	Завершение процесса
Ctrl-z	Процесс	Приостановка процесса
Ctrl-\	Процесс	Прерывание процесса с дампом памяти
Ctrl-s	Процесс	Приостановить вывод
Ctrl-q	Процесс	Продолжить вывод
Ctrl-d	Процесс	Конец потока данных
Ctrl-l	Терминал	Очистка экрана
Ctrl-_	Терминал	Отмена выполненого редактирования
Ctrl-v	Терминал	Не интерпретировать следующий символ

Символ	Объект	Действие
Enter	Строка	Конец строки
Ctrl-a	Строка	Перевести курсор на начало строки
Ctrl-e	Строка	Перевести курсор в конец строки
Alt-b	Строка	Перевести курсор на слово назад
Alt-f	Строка	Перевести курсор на слово назад
Ctrl-w	Строка	Вырезать слово слева от курсора
Ctrl-y	Строка	Извлечь из буфера
Ctrl-k	Строка	Удаление символов до конца строки
Ctrl-u	Строка	Удаление символов до начала строки

- **stty** - команда, позволяющая управлять режимами работы терминала, в том числе и обработкой управляющих символов

```
$ stty -a
```

- Отображение текущих настроек работы терминала
- **reset** - переинициализация терминала

Раздел 2

Перенаправление ввода-вывода

Примеры применения

```
$ env > variables.list
```

- Запись результатов выполнения в файл

```
$ grep "bash" < /etc/passwd
```

- Обработка данных из файла

```
$ ls /etc | nl
```

- Передача вывода другому процессу для обработки

Консольный ввод-вывод

Рисунок 2: Консольный ввод-вывод процесса

Стандартные дискрипторы ввода/вывода

ID	Имя	Описание	Перенаправление
0	STDIN	стандартный поток ввода	<
1	STDOUT	стандартный поток вывода	>(1>) / »
2	STDERR	стандартный поток ошибок	2>

- Пример

```
$ ls -l dir 2> err-list 1> corr-list
```

Получение STDIN из файла

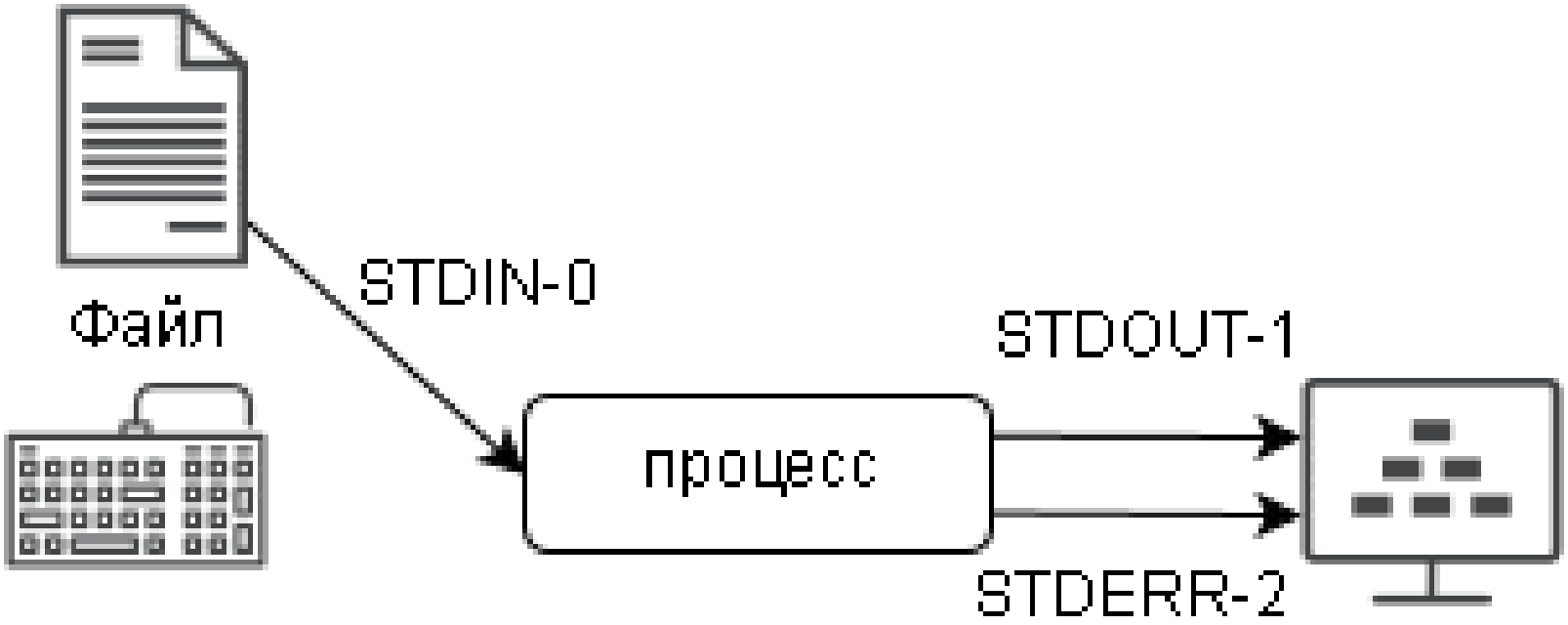

Рисунок 3: Перенаправление стандартного потока ввода процесса

Пример

```
$ sort < /etc/passwd
```

Перенаправление содержанием файла

```
$ grep world <(echo "hello world")
```

Перенаправление результатом вывода команды

Перенаправление STDOUT в файл

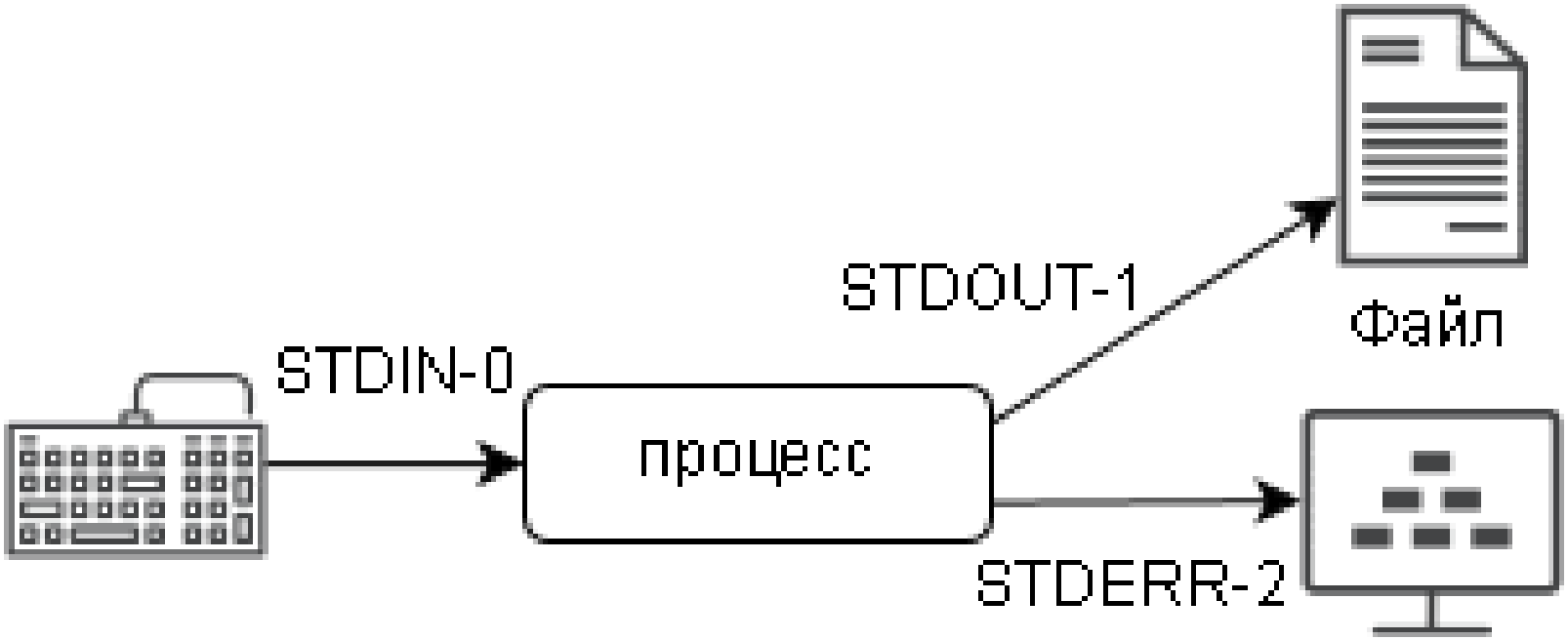

Рисунок 4: Перенаправление стандартного потока вывода в файл

Пример

```
$ echo "Line 2" >> example.txt
$ ls -l /var/log > log.files
$ tar -zc include > ~/include-backup.tar.gz
$ gzip < file.txt > file.txt.gz
$ grep -v "^#" /etc/resolv.conf > resolv2.conf
```

Перенаправление STDERR в файл

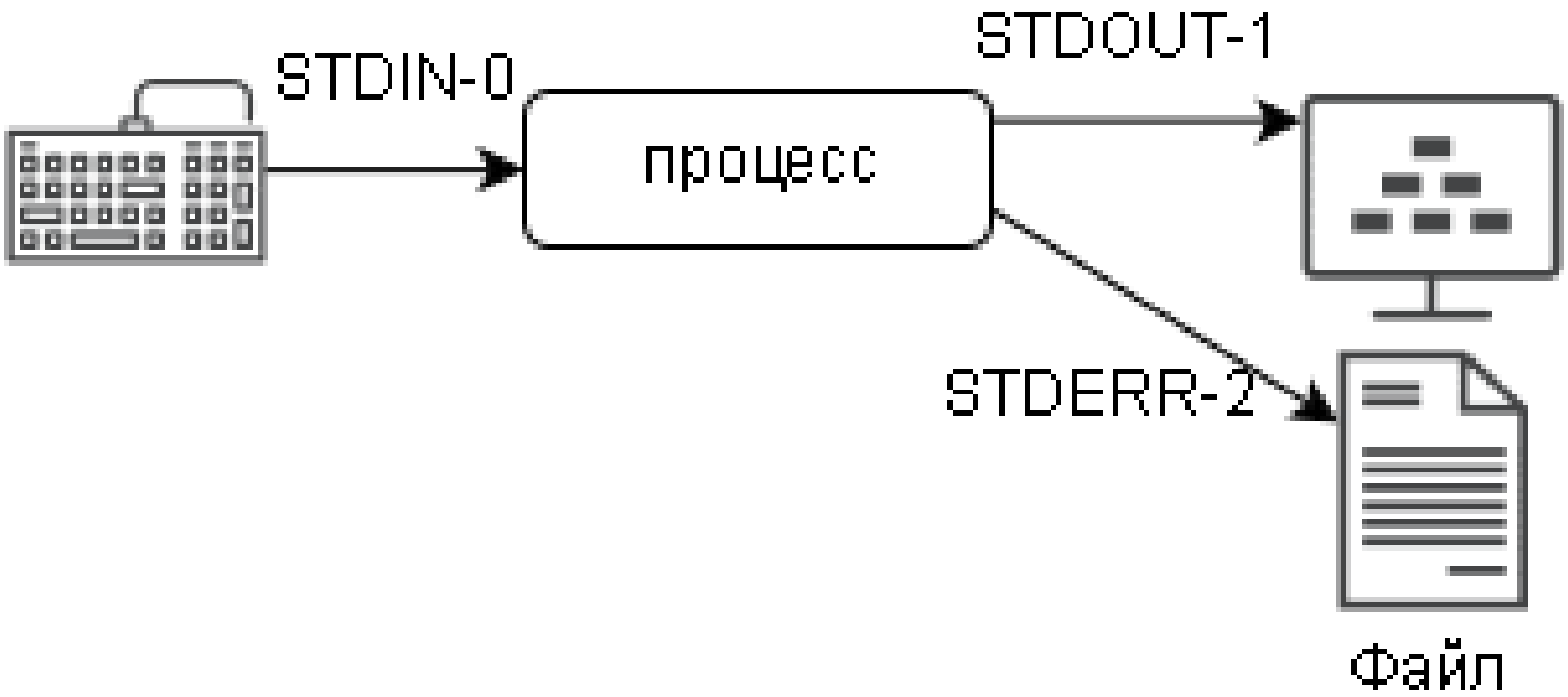

Рисунок 5: Перенаправление стандартного потока ошибок в файл

Пример

```
$ ls /fake
ls: cannot access /fake: No such file or directory
$ ls /fake > output.txt
ls: cannot access /fake: No such file or directory
$ ls /fake 2> error.txt
```

- Подавление вывода

```
$ ls /not-exist 2> /dev/null
```

вывод в никуда - подавление потока ошибок

Объединение потоков вывода

```
$ ls /fake /etc/ppp > example.txt 2> error.txt
```

*Направляем **STDOUT** и **STDERR** в разные файлы*

```
$ ls /fake /etc/ppp &> all.txt
```

*Объединяем потоки **STDOUT** и **STDERR***

```
$ find /etc > log.txt 2>&1
```

Другой синтаксис

Использование завершающих символов при перенаправлении

```
$ cat > file << EOF
> hello
> world
> EOF
```

Перенаправляется все вплоть до символа EOF

Каналы команд (Command Line Pipes)

```
$ ls /etc | head
$ ls /etc/ppp | nl
$ cat /etc/passwd | sort
$ grep vasya /etc/group | sort > groups.txt
```

- Принцип - **STDOUT** одного процесса объединяется с **STDIN** другого процессом средствами командного интерпретатора

```
$ ls /fake /etc/ppp 2>&1 | nl
```

- Передача потока ошибок в канал - требует конструкции объединения каналов

```
$ dd if=/dev/sda1 | pv --progress --rate | gzip -
9 > sda1.gz
```

- Количество объединяемых команд не ограничено.

Ветвление каналов вывода - утилита tee

- Используется, как правило в конвеерах выполнения команд, когда надо сохранять промежуточные результаты.

```
$ wc -l file1.txt | tee file2.txt
```

- Проверит количество строк в файле file1.txt, выведет результат в терминал и сохранит его в файле file2.txt.

```
$ ls | tee test.txt | grep 'py'
```

- Выведет список файлов внутри папки и с помощью первого канала запишет вывод в файл test.txt. После этого передаст вывод третьей команде — grep для идентификации файлов, содержащих в себе строку py

Раздел 3

Символы экранирования

Двойные кавычки (нестрогое экранирование)

```
$ mkdir Long Dir
$ mkdir "Long Dir"
```

- Показывают командному интерпретатору, что все что заключено в кавычки должно рассматриваться как единая строка , а не набор параметров разделенных пробелом и спецсимволов, которые должны интерпретироваться отдельно

Одинарные кавычки (строгое экранирование)

```
$ echo "PATH is $PATH"
PATH is /home/egor/bin:/usr/lib/kf5/bin:/usr/local/bin:
    /usr/bin:/bin:/usr/X11R6/bin:/usr/games
$ echo "PATH with \$ is $PATH"
PATH with $ is /home/egor/bin:/usr/local/bin:/usr/bin:
        /bin:/usr/X11R6/bin:/usr/games
$ echo 'PATH is $PATH'
PATH is $PATH
```

- Дополнительно к функционалу двойных кавычек блокируют подстановку переменных и подстановку выполнения команд

Обратная косая черта

```
$ echo "PATH is $PATH"
PATH is /home/egor/bin:/usr/lib/kf5/bin:/usr/local/bin:
    /usr/bin:/bin:/usr/X11R6/bin:/usr/games
$ echo "PATH is \$PATH"
PATH is $PATH
```

- Позволяет экранировать одиночный символ

Раздел 4

Объединение выполнения команд

Подстановка выполнения команд (command substitution)

- Примеры использования подстановки

```
$ ls -l `which date`
$ dpkg -S $(which find)
```

- Использование подстановки вмесе с экранированием

```
$  date
Mon Nov  4 03:35:50 UTC 2018
$ echo "Сегодня date"
Сегодня date
$ echo "Сегодня `date`"
Сегодня Сб ноя  2 18:21:42 MSK 2019
$ echo "Сегодня $(date)"
Сегодня Сб ноя  2 18:21:42 MSK 2019
```

Управляющие последовательности (символы объединения)

- Позволяют выполнять несколько команд разом (последовательно)
- Безусловно, или в зависимости от результата выполнения предыдущей команды

```
$ cal 1 2030; cal 2 2030; cal 3 2030
```

- Символ **;** между командами - безусловное объединение

Условные объединения

```
$ ls /etc/ppp && echo success
ip-down.d  ip-up.d
success
$ ls /etc/junk && echo success
ls: cannot access /etc/junk: No such file or directory
```

- Объединение “И” - вторая команда выполняется только при успехе выполнения первой

```
$ cd /srv/data/tmp && rm -rf *
```

- Удаление всего содержимого заданного каталога

```
$ ls /etc/ppp || echo failed
ip-down.d  ip-up.d
$ ls /etc/junk || echo failed
ls: cannot access /etc/junk: No such file or directory
failed
```

- Объединение “ИЛИ” - вторая команда выполняется только при неуспехе первой

Запуск отдельного интерпретатора для выполнения группы команд

```
$ (cd /usr/include ; tar -czf $HOME/bacup.tgz *)
$ (cd /srv/data/tmp && rm -rf *)
```

- После выполнения группы команд восстановится начальный контекст, т.к. отдельный командный интепретатор будет завершен.