Linux and more.
Bash commands and processes

Artem Trunov for Ozon Masters

Вспомним, как запускать

- `python3`
  - Процесс выводит в терминал, принимает ввод с терминала
  - Когда заканчивается, возвращает нам контроль над терминалом
  - Может быть остановлен Ctrl-z
    - bg - продолжен в фоновом режиме (background)
    - fg - возвращен в интерактив (foreground)
- `journalctl -f &`
  - Запускается фоном, мы можем продолжать работать в терминале
  - Все еще выводит в терминал
- `journalctl -f > log1 &`
  - Выводит в файл

Процессы

- Процесс - любая программа, запущенная в системе.
- Каждому процессу присваивается PID, поддерживается цепочка родства процессов.
  - процесс №1 - systemd или init
- Процессы запускаются с привилегиями (и ограничениями) пользователей, их запустивших.
- Типы процессов
  - системные демоны
  - прикладные демоны
  - программы, утилиты

Посмотреть процессы

- запуск любой команды порождает процесс, а завершение её работы уничтожает её процесс
- ps - получить информацию о процессах.

```
artem@artem-ubuntu2:~$ ps
  PID TTY          TIME CMD
  489 pts/37   00:00:00 ps
28392 pts/37   00:00:00 bash
artem@artem-ubuntu2:~$ ps
  PID TTY          TIME CMD
  492 pts/37   00:00:00 ps
28392 pts/37   00:00:00 bash
artem@artem-ubuntu2:~$ ps -p 492
  PID TTY          TIME CMD
artem@artem-ubuntu2:~$
```

- Ps без аргументов и опций показывает процессы, запущенные в данной сессии bash
- На картинке показано, что при каждом запуске ps назначается новый PID
-

Запуск команд

- Зайдите на сервер и запустите **python3**
- В другом окне зайдите на сервер с убедитесь, что команда запущена

```
datamove@linux1:~$ ps -u
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
datamove   47287  0.1  0.1  10812  5080 pts/4    Ss   09:05   0:00 -bash
datamove   47299  0.0  0.3  18284  9408 pts/4    S+   09:05   0:00 python3
datamove   47336  1.0  0.1  10812  5060 pts/5    Ss   09:05   0:00 -bash
datamove   47346  0.0  0.1  11476  3472 pts/5    R+   09:05   0:00 ps -u
```

- Принудительно закройте первое окно (мышью на крестик etc) и снова посмотрите процессы во втором окне

```
datamove@linux1:~$ ps -u
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
datamove   47336  0.1  0.1  10812  5064 pts/5    Ss   09:05   0:00 -bash
datamove   47358  0.0  0.1  11476  3392 pts/5    R+   09:05   0:00 ps -u
```

parent-child

- Дерево процессов

```
datamove@linux1:~$ pstree -pu datamove
sshd(47335)──bash(47336)──pstree(47487)

sshd(47431)──bash(47432)──python3(47442)

systemd(47254)──(sd-pam)(47261)
datamove@linux1:~$
```

- Когда вы принудительно убиваете процесс, он убивает процессы, которые породил
  - С помощью сигнала HUP (Hang UP)

nohup

- Nohup перехватывает сигнал HUP и оставляет детский процесс запущенным.
  - (его родителем становится

```
datamove@linux1:~$ nohup python3 < /dev/random
nohup: appending output to 'nohup.out'
```

- Этот детский процесс получает нового родителя - процесс №1

```
datamove@linux1:~$ ps -u datamove -o pid,ppid,command -H
    PID    PPID COMMAND
  47690   47652 sshd: datamove@pts/4
  47691   47690   -bash
  47702   47691     python3
  47335   47313 sshd: datamove@pts/5
  47336   47335   -bash
  47866   47336     ps -u datamove -o pid,ppid,command -H
  47648       1 python3
  47581       1 python3
  47254       1 /lib/systemd/systemd --user
  47261   47254   (sd-pam)
```

Как убить процесс

- **`kill`** - посылает сигнал процессу
  - По умолчанию - TERM (terminate)

```
datamove@linux1:~$ ps
    PID TTY          TIME CMD
  47691 pts/4    00:00:00 bash
  48006 pts/4    00:00:00 tail
  48079 pts/4    00:00:00 journalctl
  48080 pts/4    00:00:00 ps
datamove@linux1:~$ kill 48006
datamove@linux1:~$ ps
    PID TTY          TIME CMD
  47691 pts/4    00:00:00 bash
  48079 pts/4    00:00:00 journalctl
  48081 pts/4    00:00:00 ps
[1]-  Terminated              tail -f /var/log/wtmp
```

- Прибить цепляющийся за жизнь процесс: **`kill -9`**

Проверка кода выхода программы

```
$ ls ~/.ssh >/dev/null
$ echo $?
0

$ ls -al ку-ку
ls: cannot access 'ку-ку': No such file or directory
$ echo $?
2
$ echo $?
0

$ ls -al ку-ку 2>/dev/null
$ code=$?
```

Какие коды выхода бывают? См. man или исходники (стандарта нет)

Проверка кода выхода программы

```
if ls -al ку-ку 2>/dev/null
then echo No error
else echo Error
fi
```

If *command* выполняется в соответствии с кодом выхода программы

```
if ls -al ку-ку 2>/dev/null; then echo No Error; else echo
Error; fi
```

Проверка кода выхода программы

```
ls -al ку-ку 2>/dev/null && echo No error || echo Error
```

- Command1 && command2
  - Если команда 1 **успешна**, выполнить команду 2
- Command1 || command2
  - Если команда 1 **неуспешна**, выполнить команду 2
- Command1 && command2 || command3
  - Если команда 1 успешна, выполнить команду 2, иначе выполнить команду 3

```
ls -al ку-ку 2>/dev/null && mv ку-ку q-q || touch q-q
ls -al ку-ку 2>/dev/null && mv ку-ку q-q || touch q-q || echo Cant create

ls -al ку-ку 2>/dev/null && mv ку-ку q-q || touch q-q || echo Cant create && echo Can
```

Запуск команд в питоне

- os. system
  - Старый способ
- subprocess.run
  - Новый
- В Юпитере
  - !ls