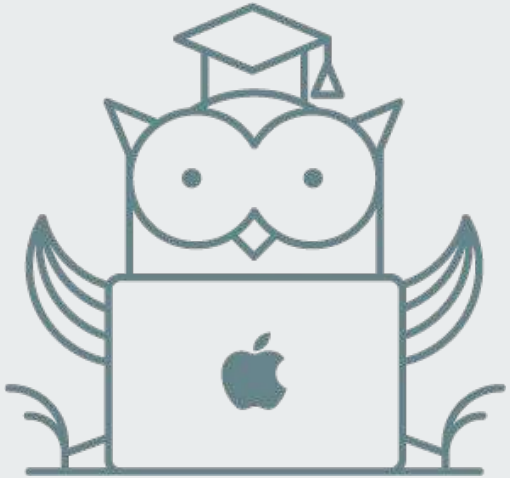
Курс «Администратор Linux»

Управление процессами

Занятие # 6

Дмитрий Молчанов
Григорий Ожегов



Процессы в linux

- Процесс и его атрибуты
- Жизнь процессов
- Получение информации о процессе

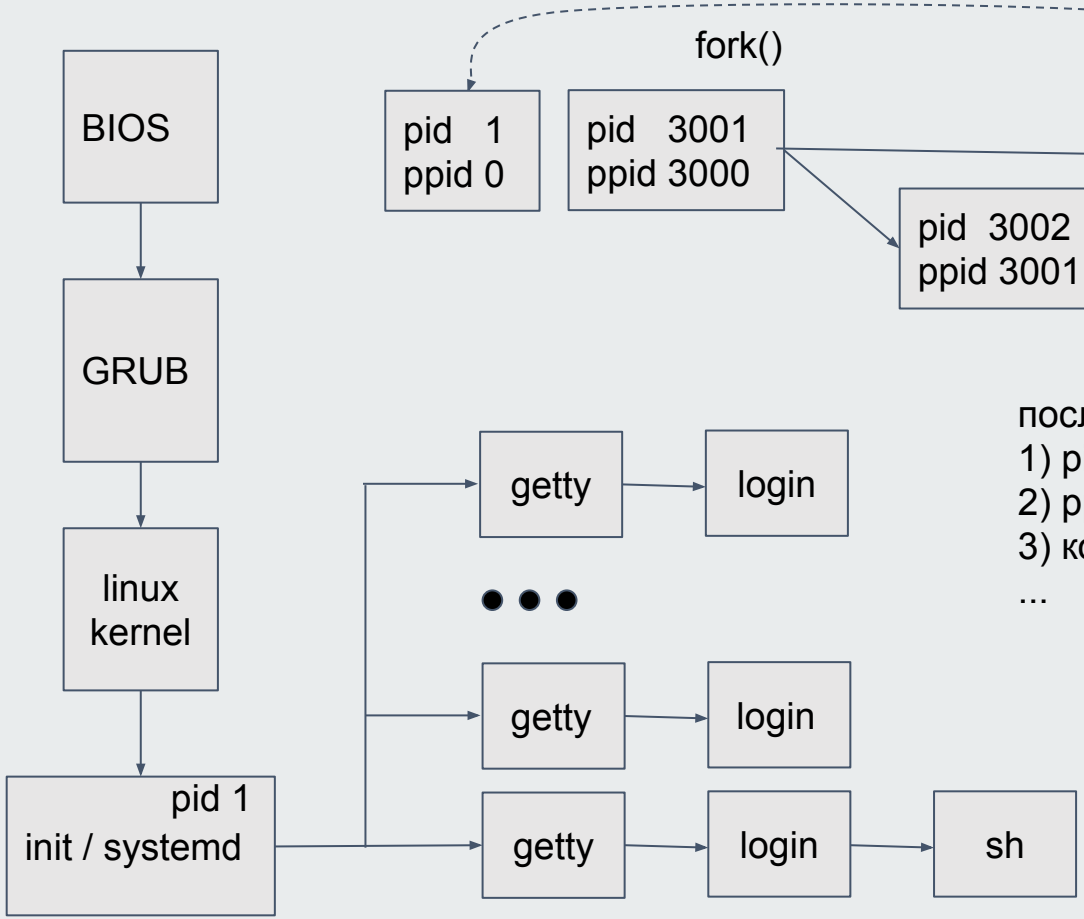
Аттрибуты процесса

- Pid - ID Процесса
- ppid - ID Процесса-родителя
- pgid - ID Группы процессов
- sid - ID сессии

Аттрибуты процесса

- Pid, ppid, pgid, sid
- State
  - R - Running
  - S - Sleeping
  - D - Uninterruptable I/O
  - Z - Zombie
  - t - Trace
  - T - STOP
- nice - приоритет планировщика (от -20 - наивысший, до +19 - низший)

Ветвление процессов



после fork() процессы отличаются:
1) pid
2) ppid
3) код возврата fork()
...

fork()

Функция fork() создает child-процесс из текущего, процессы, как бы, размножаются делением.

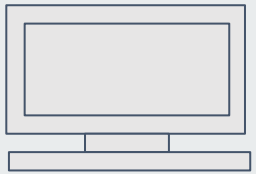
Возвращает:

child_pid - если вызван из родителя и child создался.

-1 - если произошла ошибка

0 - если вызван из child’a

Ветвление процессов



sh
pid 1000

tail -f /var/log/syslog
pid 1001

|

grep error > /tmp/log &
pid 1002

pgid 1001

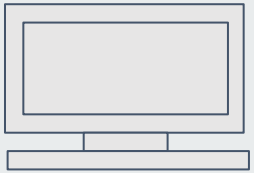
tail -f /var/log/syslog
pid 1003

|

grep error
pid 1004

pgid 1003

sid 1000



sh
pid 1100

ps ax
pid 1101

|

grep http
pid 1102

pgid 1101

sid 1100

Интерпретаторы

Любой исполняемый файл в linux запускается через свой интерпретатор/обработчик.

Для интерпретируемых скриптов интерпретатор задается через ‘shebang’ - 1ю строчку в файле начинающуюся с ‘#!’, а далее следует путь к интерпретатору.

Для бинарных файлов elf в качестве обработчика используется /lib64/ld-2.17.so. Это можно использовать для запуска бинарных файлов без атрибута eXecutable.

Процессы и сигналы

Основная масса процессов в linux - фоновые процессы никак не взаимодействующие с пользователем и не имеющие текстовых/графических интерфейсов управления или интерфейсов командной строки. Универсальный способ взаимодействие с ними - системные сигналы.

Для того, чтобы отправить сигнал процессу используют утилиту kill и один из атрибутов процесса - PID.

Процессы и сигналы

#	NAME	значение
1	HUP	HangUP - чаще всего перечитать конфигурацию
2	INT	INTerrupt received
9	KILL	незамедлительное завершение процесса
11	SEGV	SEGmentation Violation
13	PIPE	broken PIPE - запись в pipe из которого никто не читает
15	TERM	TERMination signal
19	STOP	STOP Proces

Ресурсы и состояния

Состояния процессов могут отражать какие ресурсы процессы потребляют. Так, например состояние `D` говорит, чаще всего о дисковом вводе-выводе, а состояние `R` - о потреблении `CPU`.

Получение базовой информации о процессе

- ps
- procfs
- lsof

ps

Полезные опции программы ps:

- -ef или ax - расширенный вывод обо всех процессах
- u - информация о пользователе от которого запущен процесс (включено в -ef)
- w - расширить поле cmd (ww - не ограничивать)
- f - вывод дерева процессов
- o - определить формат вывода

```
[root@linux-demo ~]# ps axfo pid,ppid,pgid,sid,stat,cmd | grep $$
 3375  3372  3375  3375 Ss        \_ -bash
 4025  3375  4025  3375 R+            \_ ps axfo pid,ppid,pgid,sid,stat,cmd
 4026  3375  4025  3375 S+            \_ grep --color=auto 3375
```

procfs: /proc/$pid

- /proc/$pid/
  - cwd - ссылка на текущий каталог процесса
  - fd/ - директория со списком
  - exe - ссылка на исполняемый файл
  - status - файл с детальной информацией о процессе
  - stat - машиночитаемый файл с информацией о процессе

Расширенная информация о процессе

Помимо того, чтобы наблюдать за процессом со стороны можно заглянуть ему под капот с помощью программ gdb и strace/ltrace.

strace/ltrace

Strace позволяет отследить какие системные вызовы использует процесс.
Для отслеживания вызовов shared библиотек используют ltrace
Полезные опции:

- -c - count statistics
- -s - ограничение длины строковых параметров выводимых на экран
- -t - вывод timestamp вызова
- -T - вывод времени затраченного на вызов
- -f - вывод информации о процессе И его потомках
- -o - вывод информации в файл
- -e - фильтрация вывода

gdb

Когда не хватает информации strace можно прибегнуть к более “суровым”/тяжелым методам.

Например, gdb -p pid или gdb /path/to/program /path/to/core и команда bt.

С помощью gdb можно сделать куда более сложные и интересные вещи. Например, поменять лимиты у работающего процесса: https://gchp.ie/updating-ulimit-on-running-linux-process/

Изменение приоритетов процесса

С помощью утилиты nice можно изменить “привлекательность” процесса для планировщика ядра. Большее значение nice уменьшает приоритет, а меньшее - увеличивает.

С помощью утилиты ionice можно изменить приоритет процесса для планировщика ввода-вывода. Есть 3 класса (1: realtime,2: best-effort, 3: idle) приоритетов и 8 уровней приоритета для классов 2 и 3 (меньше уровнь - больше приоритет).

Спасибо за внимание

Дмитрий Молчанов
Григорий Ожегов