PORTA ONE

Linux

Процессы

Содержание

- Программы и процессы
- Типы процессов
- Атрибуты процесса
- Сигналы
- Задания (jobs)
- Перенаправление ввода / вывода

Программы и процессы

- **Программа** – это совокупность файлов, которые прямо или косвенно можно загрузить в память и выполнить (исходный код, объектный код, исполняемый код…)
- **Процесс** – программа в памяти на стадии ее исполнения
  - Код программы
  - Окружение / среда исполнения
  - Информация о процессе
  - …
  - Одна программа может создавать несколько процессов
  - Каждая выполняемая программа, порождает хотя бы один процесс (внутренняя команда shell – может и не порождать)
  - Некоторые процессы существуют со старта системы
  - У каждого процесса есть родительский процесс

Типы процессов

- Системные процессы:
  - Являются частью ядра и всегда располагаются в памяти
  - Не имеют соответствующих им программ (исполняемых файлов)
  - Запускаются особым образом при инициализации ядра
  - Их исполняемый код – в ядре, поэтому могут обращаться к недоступным для обычных процессов ресурсам (код, память…)
  - В **ps(1)** «COMMAND» отображается в «[», «]»
  - Часто **init(8)** также считается системным, хотя и не часть ядра

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          2  0.0  0.0      0     0 ?        S<   14:10   0:00 [migration/0]
root          3  0.0  0.0      0     0 ?        SN   14:10   0:00 [ksoftirqd/0]
root          4  0.0  0.0      0     0 ?        S<   14:10   0:00 [watchdog/0]
root          5  0.0  0.0      0     0 ?        S<   14:10   0:00 [events/0]
root          6  0.0  0.0      0     0 ?        S<   14:10   0:00 [khelper]
root          7  0.0  0.0      0     0 ?        S<   14:10   0:00 [kthread]
root         10  0.0  0.0      0     0 ?        S<   14:10   0:00 [kblockd/0]
root         11  0.0  0.0      0     0 ?        S<   14:10   0:00 [kacpid]
root        170  0.0  0.0      0     0 ?        S<   14:10   0:00 [cqueue/0]
root        173  0.0  0.0      0     0 ?        S<   14:10   0:00 [khubd]
root        175  0.0  0.0      0     0 ?        S<   14:10   0:00 [kseriod]
```

Типы процессов

- Демоны:
  - Не интерактивные процессы (не привязаны к терминалу)
  - Запускаются обычным образом (из внешней программы)
  - Обычно запускаются скриптами (при загрузке системы)
  - Обеспечивают работу различных сервисов и подсистем ОС
  - Не связаны не с одним пользовательским сеансом работы
  - Не могут непосредственно управляться пользователями
  - Обычно, большую часть времени ожидают запросов
  - Не привязаны к конкретному TTY: «?»

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root        577  0.0  0.1   2476   932 ?        S<s  14:11   0:01 /sbin/udevd -d
root       2266  0.0  0.1  12524   756 ?        S<sl 14:11   0:00 auditd
root       2268  0.0  0.1  12080   668 ?        S<sl 14:11   0:00 /sbin/audispd
root       2300  0.0  0.1   1728   620 ?        Ss   14:11   0:00 syslogd -m 0
root       2303  0.0  0.0   1680   396 ?        Ss   14:11   0:00 klogd -x
rpc        2338  0.0  0.1   1816   548 ?        Ss   14:11   0:00 portmap
root       2375  0.0  0.1   1868   736 ?        Ss   14:11   0:00 rpc.statd
root       2409  0.0  0.1   5520   588 ?        Ss   14:11   0:00 rpc.idmapd
dbus       2434  0.0  0.1   2752   928 ?        Ss   14:11   0:00 dbus-daemon --system
```

Типы процессов

- Прикладные процессы:
  - «Остальные процессы» системы
  - Обычно привязаны к терминалу
  - Запускаются в рамках пользовательского сеанса работы
  - Время жизни, обычно, ограничено сеансом работы пользователя
  - Обычно, «монолитно» владеют терминалом (TTY)
  - При потере управляющего терминала, обычно, уничтожаются
  - Могут выполняться как в интерактивном так и фоновом режиме

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root       2931  0.0  0.0   1664   420 tty1     Ss+  14:12   0:00 /sbin/mingetty tty1
root       2932  0.0  0.0   1664   424 tty2     Ss+  14:12   0:00 /sbin/mingetty tty2
root       2933  0.0  0.0   1664   420 tty3     Ss+  14:12   0:00 /sbin/mingetty tty3
root       2944  0.0  0.0   1664   420 tty4     Ss+  14:12   0:00 /sbin/mingetty tty4
root       2947  0.0  0.0   1664   420 tty5     Ss+  14:12   0:00 /sbin/mingetty tty5
root       2950  0.0  0.0   1664   420 tty6     Ss+  14:12   0:00 /sbin/mingetty tty6
roman      2993  0.0  0.2   5524  1496 pts/0    Ss   14:14   0:00 -bash
roman      3034  0.0  0.1   5684   936 pts/0    S+   14:14   0:00 screen
roman      3036  0.0  0.3   5536  1576 pts/1    Ss   14:14   0:00 /bin/bash
roman      3128  0.0  0.2   5536  1512 pts/2    Ss+  15:05   0:00 /bin/bash
roman      3170  0.0  0.2   5536  1512 pts/3    Ss   15:05   0:00 /bin/bash
```

Атрибуты процесса

- **Process ID (PID)** – идентификатор процесса
  - Уникальный номер процесса (1-99999)
  - Присваивается на единицу больше предыдущего присвоенного
  - Циклически выбирается минимальный доступный
- **Parent PID (PPID)** – PID родительского процесса
  - PID процесса, запустившего текущий
- **NICE** – статический приоритет процесса
  - Относительный приоритет выполнения процесса
  - Учитывается планировщиком задач при определении очередности запуска процессов (предоставления процессора)

Атрибуты процесса

- **Real UID (RUID)** – реальный UID
  - Идентификатор пользователя, запустившего процесс
- **Effective UID (EUID)** – эффективный UID
  - Определяет права доступа процесса (к ресурсам системы)
  - Равен RUID, если не указан SUID флаг для исполняемого файла (тогда EUID равен UID владельца исполняемого файла)
- **Real GID (RGID)** – реальный GID
  - Идентификатор группы пользователя, запустившего процесс
- **Effective (EGID)** – эффективный GID
  - Определяет права доступа процесса (к ресурсам системы)
  - Равен RGID, если не указан SGID флаг для исполняемого файла (тогда EGID равен GID группы владельца исполняемого файла)

Атрибуты процесса

- **TTY** – терминал или псевдотерминал
  - Ассоциирован с процессом (не всегда)
  - Некоторые процессы не имеют ассоциированного терминала
  - Связывает стандартные потоки ввода/вывода
- Другие атрибуты:
  - Дополнительные группы
  - Umask – маска режима создания файлов и директорий
  - Limits – ограничения процесса на использование ресурсов
  - Root – рабочая директория процесса
  - …

Мониторинг процессов

PORTA ONE

ps - моментальный слепок состояния процессов в системе

- **w** В широком формате, без усечения до размера экрана
- **a** Всех пользователей
- **x** Не зависимо от того, связаны ли они с терминалом (в том числе демоны)
- **o fields** Задать поля для отображения
- **j, l, u, v** Варианты наборов полей для отображения

 # ps
 # ps uaxwww
 # ps axo pid,command

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.1   2072   624 ?        Ss   14:10   0:02 init [3]
root          2  0.0  0.0      0     0 ?        S<   14:10   0:00 [migration/0]
root          3  0.0  0.0      0     0 ?        SN   14:10   0:00 [ksoftirqd/0]
root          4  0.0  0.0      0     0 ?        S<   14:10   0:00 [watchdog/0]
root          5  0.0  0.0      0     0 ?        S<   14:10   0:00 [events/0]
root          6  0.0  0.0      0     0 ?        S<   14:10   0:00 [khelper]
root          7  0.0  0.0      0     0 ?        S<   14:10   0:00 [kthread]
root         10  0.0  0.0      0     0 ?        S<   14:10   0:00 [kblockd/0]
```

top - мониторинг процессов во времени

- **f** Выбор полей для отображения, выбор поля для сортировки
- **s** Задать время обновления данных

```
top - 21:25:22 up  7:14,  2 users,  load average: 0.00, 0.00, 0.00
Tasks:  90 total,   1 running,  89 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.1%us,  0.6%sy,  0.0%ni, 98.9%id,  0.2%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    515340k total,   281696k used,   233644k free,    34348k buffers
Swap:  1048568k total,        0k used,  1048568k free,   188184k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 6113 root      15   0  2332  896  696 R  1.2  0.2   0:00.03 top
    1 root      15   0  2072  624  532 S  0.0  0.1   0:02.73 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
```

Сигналы

- **Сигналы** – это элемент межпроцессного взаимодействия, позволяющий передавать от одного процесса (или ядра ОС) другому уведомления о возникновении определенного события

- **Отправка сигнала:**
  - Согласно прав пользователя (EUID)
  - Утилитами **kill(1)**, **killall(1)**, **pkill(1)**
  - Пользователем с клавиатуры (“stty -a” - список всех комбинаций)
  - Ядром или другим процессом

- **Получение / обработка сигнала**:
  - Действие по умолчанию (обычно – завершение процесса)
  - Игнорировать (кроме KILL и STOP)
  - Перехватить и обработать (кроме KILL и STOP)

Сигналы

- **1 HUP (Отбой)** – перечитать конфигурацию (без останова сервиса), перезапуск, выход, *завершение*
- **2 INT (Прерывание)** – выход, *завершение* (^C)
- **3 QUIT (Выход)** – выход, *завершение* (редко)
- **9 KILL (Убить)** – не игнорируемое/не обрабатываемое *уничтожение* (когда другие варианты уже не помогают) – не корректное завершение
- **14 ALRM (Сигнал тревоги)** – срабатывание таймера, *завершение*
- **15 TERM (Завершение)** – нормальное *завершение* работы
- **STOP (Останов)** – *остановить* выполнение процесса
- **CONT (Продолжить)** – возобновить работу процесса
- **USR1 / USR2 (Пользовательские)** – определен пользователем

Утилиты для отправки сигналов

kill - послать сигнал процессу (по умолчанию TERM)

kill [-s имя_сигнала] pid ...

kill -l – отобразить все поддерживаемые сигналы

pid - Идентификатор процесса (может быть несколько)

n	Отправить сигнал указанному процессу
-n	Отправить сигнал всем процессам указанной группы процессов(**n>1**)
0	Отправить сигнал всем процессам текущей группы процессов

 # kill -s HUP 507

 # kill 142 157

killall — послать сигнал группе процессов по имени

```
killall [-wdivrg] [-u user] [-t term] [-SIG] [-s SIG] [name ...]
```

-i Интерактивно

-r Интерпретировать **name** как **regexp**

-u user Только процессы, принадлежащие указанному пользователю

-w Ожидать завершения процессов

-s SIG Отправить указанный сигнал

```
$ killall -s 1 man
# killall -HUP daemond
```

Приоритет выполнения

nice - запустить программу с измененным приоритетом

nice [-n приращение] программа [аргумент ...]

Чем меньше значение nice у процесса, тем более высокий приоритет он имеет.

-n - приращение, увеличить приоритет на указанное приращение (по умолчанию – 10)

 # /usr/bin/nice –n -10 top

$ nice –n 20 grep USER /var/log/message > USER.log

renice - изменить приоритет программы уже после её запуска (в процессе её выполнения)

`renice приоритет [[-p] pid ...] [[-g] pgrp ...] [[-u] user ...]`

По умолчанию аргумент интерпретируется как pid.

приоритет Изменение приоритета (+/-n)

-p pid Изменить для перечисленных PID

-u user Изменить для перечисленных пользователей (UID)

```
# renice +1 987 -u daemon root -p 32
$ renice +10 1234
```

Задания (jobs)

- Каждая **программа** которую запускает пользователь в рамках оболочки называется **заданием**
- Задания поддерживаются командным интерпретатором (подсистема управления заданиями)
- Каждое задание при запуске получает уникальный номер – **идентификатор задания**

- Состояния заданий:
  - Выполнение в текущем режиме (foreground)
  - Выполнение в фоновом режиме (background)
  - Выполнение приостановлено (stopped)

Идентификация заданий

%N	Номер задания
%S	Вызов (командная строка) задания, которая начинается со строки **S**
%?S	Вызов (командная строка) задания, которая содержит строку **S**
%+	«текущее» задание (последнее задание приостановленное на переднем плане или запущенное в фоне), аналогично записи %%
%-	Последнее задание
$!	Последний фоновый процесс

Команды управления заданиями

- **jobs** – выводит список заданий, исполняющихся в фоне

  # jobs

- **fg** – переводит задание из фона на передний план
  - **# fg %1**
  - **# fg %-**
- **bg** – перезапускает приостановленное задание в фоновом режиме

  # bg

- **wait** – ожидает завершения работы задания/процесса (возвращает код завершения задания/процесса)

  # wait # wait %4 # wait 11234

- **kill** – отправляет сигнал заданию/процессу

  # kill %2 # kill -1 1235

Перенаправление I/O

PORTA ONE

- STDIN – стандартный ввод
  По умолчанию – клавиатура терминала
  Файловый дескриптор – 0
- STDOUT – стандартный вывод
  По умолчанию – экран терминала
  Файловый дескриптор – 1
- STDERR – стандартный вывод ошибок
  По умолчанию – экран терминала
  Файловый дескриптор – 2

Стандартные потоки – потоки процесса, имеющие номер (дескриптор) зарезервированный для выполнения некоторых стандартных функций (обычно уже открыты при запуске)

Файловый дескриптор – неотрицательное целое число, присваиваемое ОС открытому процессом файлу

Перенаправление I/O

Синтаксис	Описание перенаправления
cmd >file	STDOUT в файл file, с предварительной очисткой содержимого (или созданием)
cmd >>file	STDOUT в файл file, с добавлением в конец (или созданием)
cmd <file	Из существующего файла file в STDIN
cmd <<END	С клавиатуры в STDIN до маркера END
cmd n>file	Из файлового дескриптора n в файл file, с предварительной очисткой содержимого (или созданием)
cmd n>>file	Из файлового дескриптора n в файл file, с добавлением в конец (или созданием)
cmd &>file	STDOUT и STDERR в файл file, с предварительной очисткой содержимого (или созданием)
cmd n>&m	Из файлового дескриптора n в файловый дескриптор m

Перенаправление I/O

Конвейеры (pipes) – это возможность нескольких программ работать совместно, перенаправляя стандартный вывод одной (STDOUT) на стандартных ввод (STDIN) другой

Синтаксис	Описание конвейеризации
cmd | cmd2	Передача STDOUT команды cmd в STDIN команды cmd2

```
$ echo $(systemctl list-units --type=target | grep custom | awk '{print $1 $2 $3}') $(systemctl list-dependencies custom.target | head -3 | sed s/[^a-z]//g | sort) | md5sum | cut -c 10-16
```