

## Лабораторная работа №2. Мониторинг процессов и ресурсов в ОС Linux

### Рассматриваемые вопросы

1. Получение информации о запущенных процессах
2. Получение информации об используемых процессами ресурсах
3. Представление результатов в удобном для анализа и обработки виде

### Идентификация процессов

Система идентифицирует процессы по уникальному номеру, называемому идентификатором процесса или **PID** (process ID).

Все процессы, работающие в системе GNU/Linux, организованы в виде дерева. Корнем этого дерева является **init** или корневой процесс **systemd** – процессы системного уровня, запускаемые во время загрузки. Для каждого процесса хранится идентификатор его родительского процесса (**PPID**, Parent Process ID). Для корневого процесса в качестве идентификатора родительского процесса указывается 0.

### Получение общих сведений о запущенных процессах

Команда **ps** (сокращение от process status)

Запуск **ps** без аргументов покажет только те процессы, которые были запущены Вами и привязаны к используемому Вами терминалу.

Часто используемые параметры (указываются без "-"):

- a** – вывод процессов, запущенные всеми пользователями;
- x** – вывод процессов без управляющего терминала или с управляющим терминалом, но отличающимся от используемого Вами;
- u** – вывод для каждого из процессов имя запустившего его пользователя и времени запуска.

Обозначения состояний процессов (в колонке **STAT**)

- R** – процесс выполняется в данный момент или находится в состоянии готовности;
- S** – процесс ожидает события (прерываемое ожидание);
- D** – процесс ожидает ввода/вывода (непрерываемое ожидание);
- I** – простаивающий поток ядра;
- Z** – zombie-процесс;
- T** – процесс остановлен.

Команда **pstree**

Команда **pstree** выводит процессы в форме дерева: можно сразу увидеть родительские процессы.

Часто используемые параметры:

- p** – вывод **PID** всех процессов;
- c** – развернуть все ветви;
- u** – вывод имени пользователя, запустившего процесс.

Команда **top**

**top** – программа, используемая для наблюдения за процессами в режиме реального времени. Полностью управляется с клавиатуры. Вы можете получить справку, нажав на клавишу **h**. Наиболее полезные команды для мониторинга процессов:

- M** – эта команда используется для сортировки процессов по объему занятой ими памяти (поле **%MEM**);
- P** – эта команда используется для сортировки процессов по занятому ими процессорному времени (поле **%CPU**). Это метод сортировки по умолчанию;
- U** – эта команда используется для вывода процессов заданного пользователя. **top** спросит у вас его имя. Вам необходимо ввести имя пользователя, а не его UID. Если вы не введете никакого имени, будут показаны все процессы;
- i** – по умолчанию выводятся все процессы, даже спящие. Эта команда обеспечивает вывод информации только о работающих в данный момент процессах (процессы, у которых поле **STAT** имеет значение **R**, Running). Повторное использование этой команды вернет Вас назад к списку всех процессов.

### Получение детальных сведений о запущенных процессах

**/proc** – псевдо-файловая система, которая используется в качестве интерфейса к структурам данных в ядре. Большинство расположенных в ней файлов доступны только для чтения, но некоторые файлы позволяют изменять переменные ядра.

Каждому запущенному процессу соответствует подкаталог с именем, соответствующим идентификатору этого процесса (его **PID**). Каждый из этих подкаталогов содержит следующие псевдо-файлы и каталоги (указаны наиболее часто используемые для мониторинга процессов). **Внимание!** Часть из этих файлов доступна только в директориях процессов, запущенных от имени данного пользователя или при обращении от имени **root**.

**cmdline** – файл, содержащий полную командную строку запуска процесса.

**cwd** – ссылка на текущий рабочий каталог процесса.

**environ** – файл, содержащий окружение процесса. Записи в файле разделяются нулевыми символами, и в конце файла также может быть нулевой символ.

**exe** – символьная ссылка, содержащая фактическое полное имя выполняемого файла.

**fd** – подкаталог, содержащий одну запись на каждый файл, который в данный момент открыт процессом. Имя каждой такой записи соответствует номеру файлового дескриптора и является символьной ссылкой на реальный файл. Так, **0** – это стандартный ввод, **1** – стандартный вывод, **2** – стандартный вывод ошибок и т. д.

**io** – файл, содержащий сведения об объемах данных, прочитанных и записанных процессом в хранилище.

**maps** – файл, содержащий адреса областей памяти, которые используются программой в данный момент, и права доступа к ним.

**sched** – файл, содержащий значения переменных для каждого процесса, использующихся планировщиком **CFS** для принятия решения о выделении процессу процессорного времени. Например, **sum\_exec\_runtime**, это переменная, содержащая оценку суммарного времени выполнения процесса, а **nr\_switches** – переменная, хранящая количество переключений контекста.

**stat** – машиночитаемая информация о процессе в виде набора полей;

**status** – предоставляет большую часть информации из **stat** в более лёгком для прочтения формате.

**statm** – предоставляет информацию о состоянии памяти в страницах как единицах измерения. Список полей в файле:

<b>size</b>	общий размер программы
<b>resident</b>	размер резидентной части
<b>shared</b>	количество разделяемых страниц
<b>text</b>	текст (код)
<b>lib</b>	библиотеки (не используется начиная с ядра 2.6)
<b>data</b>	данные + стек
<b>dt</b>	"грязные" (dirty) страницы (не используется начиная с ядра 2.6)

### Обработка данных о процессах

Обработка данных о процессах проводится, как правило, в рамках организации конвейера команд обработки текстовых потоков и (или) через циклическую обработку строк файлов. Советуем применять команды, изученные в рамках второй лабораторной работы – **grep**, **sed**, **awk**, **tr**, **sort**, **uniq**, **wc**, **paste**, а также функции для работы со строками.

### Задание на лабораторную работу

1. Создайте свой каталог в директории **/home/user/**. Все скрипты и файлы для вывода результатов создавайте внутри этого каталога или его подкаталогов. (**mkdir lab2**)
2. Напишите скрипты, решающие следующие задачи:
  - i) Посчитать количество процессов, запущенных пользователем **user**, и вывести в файл получившееся число, а затем пары **PID:команда** для таких процессов.
  - ii) Вывести в файл список **PID** всех процессов, которые были запущены командами, расположенными в **/sbin/**
  - iii) Вывести на экран **PID** процесса, запущенного последним (с последним временем запуска).
  - iv) Для всех зарегистрированных в данный момент в системе процессов определить среднее время непрерывного использования процессора (**CPU\_burst**) и вывести в один файл строки **ProcessID=PID : Parent\_ProcessID=PPID : Average\_Running\_Time=ART**. Значения **PPid** взять из файлов **status**, которые находятся в директориях с названиями, соответствующими **PID** процессов в **/proc**. Значения **ART** получить, разделив значение **sum\_exec\_runtime** на **nr\_switches**, взятые из файлов **sched** в этих же директориях. Отсортировать эти строки по идентификаторам родительских процессов.
  - v) В полученном на предыдущем шаге файле после каждой группы записей с одинаковым идентификатором родительского процесса вставить строку вида **Average\_Sleeping\_Children\_of\_ParentID=N is M**, где **N = PPID**, а **M** – среднее, посчитанное из **ART** для всех процессов этого родителя.
  - vi) Используя псевдофайловую систему **/proc** найти процесс, которому выделено больше всего оперативной памяти. Сравнить результат с выводом команды **top**.
  - vii) Написать скрипт, определяющий три процесса, которые за 1 минуту, прошедшую с момента запуска скрипта, считали максимальное количество байт из устройства хранения данных. Скрипт должен выводить **PID**, строки запуска и объем считанных данных, разделенные двоеточием.
3. Предъявите скрипты преподавателю и получите вопрос или задание для защиты лабораторной работы.

Доп. задание: напишите скрипт, который рекурсивно выведет все поколения (дети, внуки, правнуки, ...) процесса.