Лабораторная работа №3.
Мониторинг процессов в ОС Linux

Рассматриваемые вопросы

1. Получение информации о запущенных процессах
2. Получение информации об используемых процессами ресурсах
3. Представление результатов в различном 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** – процесс системного уровня, запускаемый во время загрузки. Для каждого процесса хранится идентификатор его родительского процесса (**PPID, Parent Process ID**). **init** сам себе является родителем – его **PID** и **PPID** равны **1**.

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

S – процесс ожидает (т.е. спит менее 20 секунд)

I – процесс бездействует (т.е. спит больше 20 секунд)

D – процесс ожидает ввода/вывода (или другого недолгого события), непрерываемый

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

-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 использующиеся 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.
maps – файл, содержащий адреса областей памяти, которые используются программой в данный момент, и права доступа к ним. Формат файла следующий:

```
address           perms offset  dev   inode      pathname
08048000-08056000 r-xp 00000000 03:0c 64593      /usr/sbin/gpm
08056000-08058000 rw-p 0000d000 03:0c 64593      /usr/sbin/gpm
08058000-0805b000 rwxp 00000000 00:00 0
40000000-40013000 r-xp 00000000 03:0c 4165       /lib/ld-2.2.4.so
bffff000-c0000000 rwxp 00000000 00:00 0
```

где **address** -- адресное пространство, занятое процессом; **perms** -- права доступа к нему:

- **r** = можно читать
- **w** = можно писать
- **x** = можно выполнять
- **s** = можно использовать несколькими процессами совместно
- **p** = личная (копирование при записи);

offset -- смещение в файле, **dev** -- устройство (старший номер : младший номер); **inode** -- индексный дескриптор на данном устройстве: **0** означает, что с данной областью памяти не ассоциированы индексные дескрипторы;

stat – детальная информация о процессе в виде набора полей;
status – предоставляет бóльшую 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:

- **size** общий размер программы
- **resident** размер резидентной части
- **share** разделяемые страницы
- **trs** текст (код)
- **drs** данные/стек
- **lrs** библиотека
- **dt** "дикие" (dirty) страницы

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 lab3**)
2. Напишите скрипты, решающие следующие задачи:
   i) Посчитать количество процессов, запущенных пользователем **user**, и вывести в файл пары **PID:команда** для таких процессов.
   ii) Вывести на экран **PID** процесса, запущенного последним (с последним временем запуска).
   iii) Вывести в файл список **PID** всех процессов, которые были запущены командами, расположенными в **/sbin/**
   iv) Для каждого процесса посчитать разность резидентной и разделяемой части памяти процесса (в страницах). Вывести в файл строки вида **PID:разность**, отсортированные по убыванию этой разности.
   v) Для всех зарегистрированных в данный момент в системе процессов выведите в один файл строки **ProcessID=PID : Parent_ProcessID=PPID : Average_Sleeping_Time=SleepAVG.** Значения **PPid** и **Pid** возьмите из файлов **status**, значение **SleepAVG** из файла **sched** поле **avg_atom,** которые находятся в директориях с названиями, соответствующими **PID** процессов в **/proc** . Отсортируйте эти строки по идентификаторам родительских процессов.
   vi) 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 **SleepAVG** для данного процесса.
3. Предъявите скрипты преподавателю и получите вопрос или задание для защиты лабораторной работы.
4. После защиты лабораторной работы удалите созданный каталог со всем его содержимым (**rm –R lab3**)