# Лабораторная работа 3. Управление процессами

**Аннотация:** Цель работы: изучить команды, позволяющие получать информацию о процессах, устанавливать их приоритеты, отправлять сигналы процессам, а также запускать и обрабатывать фоновые процессы.

**Задание 1**. Выведите список всех процессов в системе, запущенных в данный момент.

Ход выполнения:

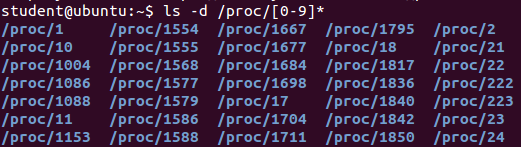
1. Для организации управления процессами в Linux существует псевдо-файловая (или виртуальная) система процессов – /proc, в которой ядро хранит информацию о процессах. Файловая система процессов используется для получения информации о запущенных процессах.

Каталог /proc выглядит как часть общей каталоговой структуры, но фактически хранится в памяти, а не на диске.

Выведете список содержимого каталога /proc, вы увидите много каталогов, именами которых являются номера. Эти каталоги содержат информацию обо всех процессах в системе, запущенных в данный момент:

*ls -d /proc/[0-9]\**

Результат выполнения команды на экране – рис. 1.



*Рис. 1.* Список содержимого каталога /proc

Обратите внимание, если вы войдете под учетной записью, не имеющей права администратора, то сможете вывести информацию только о своих собственных процессах, но не других пользователей.

2. Посмотрите, какая информация доступна для процесса 1, которым является процесс *init* и который отвечает за запуск всех остальных процессов.

3. Ознакомьтесь с основными файлами процесса 1. Вот краткое описание некоторых из них:

* *cmdline*: этот (псевдо-) файл содержит полную командную строку, использованную для вызова процесса.
* *cwd*: эта символическая ссылка указывает на текущий рабочий каталог процесса.
* *environ*: этот файл содержит все переменные окружения, определённые для этого процесса, в виде ПЕРЕМЕННАЯ=значение.
* *exe*: эта символическая ссылка указывает на исполняемый файл, соответствующий запущенному процессу.
* *fd*: этот подкаталог содержит список файловых дескрипторов, открытых в данный момент процессом.
* *maps*: когда вы выводите содержимое этого именованного канала (при помощи команды *cat*, например), вы можете увидеть части адресного пространства процесса, которые в текущий момент распределены для файла. Вот эти поля (слева направо): адресное пространство, связанное с этим распределением; права доступа, связанные с этим распределением; смещение от начала файла, где начинается распределение; старший и младший номера (в шестнадцатиричном виде) устройства, на котором находится распределенный файл; номер inode файла; и, наконец, имя самого файла. Если устройство обозначено как 0 и отсутствует номер *inode* или имя файла – это анонимное распределение.
* *root*: эта символическая ссылка указывает на корневой каталог, используемый процессом. Обычно это будет /.
* *status*: этот файл содержит разнообразную информацию о процессе: имя исполняемого файла, его текущее состояние, его PID и PPID, его реальные и эффективные UID и GID, его использование памяти и другие данные.

4. В корневом каталоге файловой системы /proc также содержатся файлы, в которых хранится информация, относящаяся к параметрам ядра (см. табл. 6.1). Посмотрите содержимое этих файлов.

Таблица 6.1. Некоторые файлы из каталога /proc

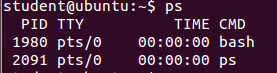
|  |  |
| --- | --- |
| cmdline | Аргументы, переданные ядру Linux при загрузке. |
| kcore | Файл, отображающий физическую память системы. |
| interrupts | Счетчики количества прерываний IRQ в архитектуре i386. |
| modules | Список модулей, загруженных системой. |
| pci | Полный список всех PCI-устройств, найденных во время инициализации ядра, а также их конфигурация. |
| version | Эта строка идентифицирует версию текущего ядра |

**Задание 2**. Отобразите список всех процессов, которые выполняются в данный момент в виде таблицы и в виде дерева.

Ход выполнения:

1. Ознакомьтесь с описанием команды *ps*.

Команда *ps* предназначена для получения информации о существующих в операционной системе процессах. Запуск *ps* без аргументов покажет только те процессы, что были запущены вами, и которые привязаны к используемому вами терминалу (рис. 6.2).



*Рис. 6.2*. Процессы, привязанные к используемому терминалу.

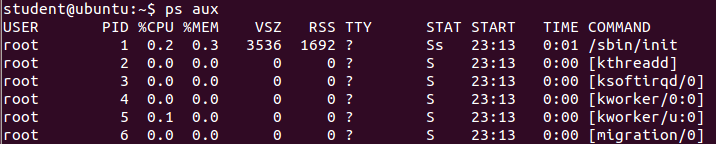
Утилита *ps* обладает рядом полезных опций, наиболее общими из которых являются:

* *a*: выводит процессы, запущенные всеми пользователями;
* *x*: выводит процессы без управляющего терминала или с управляющим терминалом, но отличающимся от используемого вами;
* *u*: выводит для каждого из процессов имя запустившего его пользователя и время запуска.

2. Введите в терминале следующую команду:

ps aux

Изучите таблицу процессов, которая появится в терминале (рис. 3).



*Рис. 3.* Вывод более подробной информации о процессах.

Вывод *ps* разделен на несколько полей. Наиболее важные из них:

* USER – показывает, от имени какого пользователя был запущен данный процесс.
* PID – уникальный идентификатор процесса
* %CPU – процессорное время, выделенное данному процессу.
* %MEM – оперативная память, выделенная данному процессу.
* TTY – терминал, на котором выполняется процесс.
* STAT – состояние процесса, например, R – исполняющийся процесс, S – спящий, Z – «зомби».
* START – время запуска программы.
* TIME – показывает, сколько процессорного времени выполняется процесс.
* COMMAND – имя программы.

3. Ознакомьтесь с описанием команды *pstree*.

Команда *pstree* выводит процессы в форме дерева. Основным преимуществом является то, что вы сразу можете увидеть родительские процессы: если вам нужно уничтожить целую серию процессов, а они все происходят от одного родителя, вы можете просто убить этот родительский процесс. Вам придется воспользоваться опцией *-p* для вывода PID всех процессов и опцией *-u* для вывода имени пользователя, запустившего процесс.

4. Запустите *pstree* следующим образом:

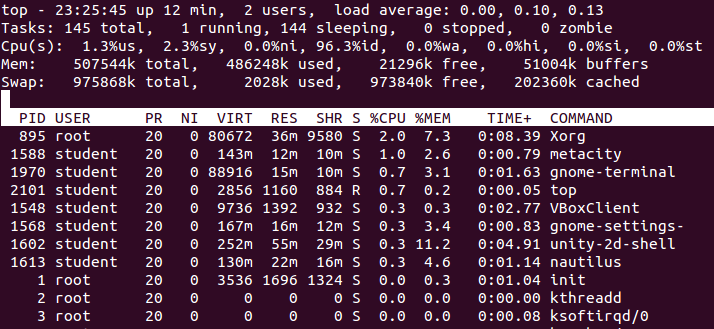
*pstree -up | more*

При этом вы получите обзор всей структуры дерева процессов.

**Задание 3**. Ознакомьтесь с информацией о текущих процессах в «реальном режиме времени».

Ход выполнения:

1. Запустите утилиту *top* (рис. 4).



*Рис. 4.* Отображение процессов в «реальном режиме времени»

2. Ознакомьтесь с тем, какую информацию можно получить о запущенных процессах в «реальном режиме времени».

Утилита *top* выводит на экран список процессов в системе, отсортированных в порядке убывания значений используемых ресурсов.

Сначала идет общесистемная информация: время запуска операционной системы, время ее работы от момента последнего перезапуска системы, количество зарегистрированных в данный момент в операционной системе пользователей, а также минимальная, максимальная и средняя загрузка операционной системы. Кроме того, отображается общее число процессов и их состояние, сколько процентов ресурсов системы занимают пользовательские процессы и системные процессы, использование оперативной памяти и свопа.

Для каждого процесса указаны идентификатор, имя владельца, приоритет, размер, состояние процесса, потребляемая оперативная память, ресурс центрального процесса, время выполнения, имя процесса.

Таблица. Назначение столбцов утилиты top

|  |  |
| --- | --- |
| **Столбец в top** | **Назначение** |
| **PID** | Идентификатор процесса, уникальное положительное целое число, которое идентифицирует процесс. |
| **USER** | Это «эффективное» имя пользователя (которое сопоставляется с идентификатором пользователя), запустившего процесс. Linux назначает действительный идентификатор пользователя и эффективный идентификатор пользователя процессам; последний позволяет процессу действовать от имени другого пользователя. |
| **NI** | «Хорошее» значение процесса. *NI*- всего лишь подсказка для ядра, какое значение приоритета процесс должен иметь. |
| **PR** | Приоритет планирования процесса с точки зрения ядра. В большинстве случаев значение *PR*может быть вычислено по следующей формуле: *PR = 20 + NI*. Теоретически ядро может самостоятельно изменить значение *PR*(но не *NI*). |
| **VIRT** | Общий объем памяти, потребляемой процессом. |
| **RES** | Память, используемая процессом в ОЗУ. |
| **SHR** | Объем памяти VIRT, который используется процессом совместно с другими процессами, например – файлы библиотек. |
| **%MEM** | Память, используемая процессом в ОЗУ, в процентах от общего объема доступной ОЗУ. |
| **S** | Состояние процесса в однобуквенной форме: R (running) – выполняется; S (sleeping) – “спящий” процесс (состояние ожидания); Z (zombie) – процесс-“зомби”. |
| **TIME+** | общее время ЦП, использованное процессом с момента его запуска, с точностью до сотых долей секунды. |
| **COMMAND** | Имя процесса. |

3. Утилита *top* полностью управляется с клавиатуры. Вы можете получить справку, нажав на клавишу *h*. Наиболее полезные команды:

* *k*: эта команда используется для отправки сигнала *kill* процессу.
* *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);
* *u*: эта команда используется для вывода процессов заданного пользователя. *top* спросит у вас его имя. Вам необходимо ввести имя пользователя, а не его UID;
* *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.
* *r*: эта команда используется для изменения приоритета выбранного процесса.

4. Отсортируйте процессы по занятому ими процессорному времени.

5. Отсортируйте процессы по объему занятой ими памяти.

**Задание 4**. Произведите остановку заранее запущенного вами любого процесса (программы).

Ход выполнения:

1. Ознакомьтесь с командами *kill* и *killall*.

Эти две команды используются для отправки сигналов процессам. Для команды *kill* требуется номер процесса в качестве аргумента, а для *killall* требуется имя процесса.

Обе эти команды допускают опциональное использование аргумента с номером сигнала, отправляемого процессу. По умолчанию они обе отправляют соответствующим процессам сигнал 15 (TERM). Например, если вам нужно убить процесс с PID 785, используйте команду:

*kill 785*

Если вам нужно отправить ему сигнал 19 (STOP), введите:

*kill -19 785*

Если вам нужно убить процесс, для которого вы знаете имя команды, вы можете убить его по имени. Если имя процесса «mozilla», вы можете воспользоваться командой:

*killall -9 mozilla*

В любом случае вы убьете только свои собственные процессы (только если вы не root), поэтому вам не стоит волноваться о процессах других пользователей, если работаете в многопользовательской системе, так как на них это не повлияет.

2. Остановите заранее запущенный вами процесс по его идентификатору.

3. Остановите заранее запущенный вами процесс по его имени.

**Задание 5**. Изучите команды установки и изменения приоритета процесса.

Ход выполнения:

1. Ознакомьтесь с утилитой *nice*.

Утилита *nice* позволяет запустить команду с предопределенным приоритетом выполнения, который задается в командной строке. При обычном запуске все задачи имеют один и тот же приоритет, и операционная система равномерно распределяет между ними процессорное время. Однако с помощью утилиты *nice* можно понизить приоритет какой-либо задачи, таким образом предоставляя другим процессам больше процессорного времени. Повысить приоритет той или иной задачи имеет право только пользователь root. Синтаксис *nice* следующий:

*nice -число команда*

Число – параметр, большее значение которого означает меньший приоритет процесса.

2. Определите, с каким приоритетом запускается программа *top*.

3. Уменьшите приоритет программы top при следующем запуске на 10.

4. Ознакомьтесь с утилитой *renice*.

Утилита *renice* позволяет изменить приоритет уже работающего процесса. Формат запуска утилиты:

*renice число PID*

5. Измените приоритет у одного из уже запущенных процессов.

**Задание 6**. Изучите команду запуска процессов в заранее определенное время.

Ход выполнения:

1. Утилита *at* позволяет создавать очередь команд для выполнения в более позднее время. Чтобы выполнить утилиту создающую сводку использования диска, называемую *du*, в 8:40 вечера выполните команду:

*at 20:40*

2. Команда *at* выведет приглашение, такое как «*at>*», где нужно ввести команду *du* с нужными параметрами:

*at> du –a >/tmp/du.out*

Здесь вывод *du* направлен в файл.

3. После ввода команды, *at* снова выведет приглашение к вводу. Чтобы выйти, надо нажать комбинацию клавиш *CTRL+D*. Появится сообщение вида:

*at> <EOT>*

warning: commands will be executed using /bin/sh

job 1 at 2013-03-12

4. Чтобы проверить, что задание находится в очереди, выполните команду *atq*. Она покажет запланированные на данный момент задания очереди. Все они хранятся в каталоге */var/spool/at.*

5. Отмените задание с помощью команды *atrm*. Для этого нужно использовать последовательный номер задания. Чтобы узнать номер задания выполните *atq*. Например, чтобы удалить задание с номером 1, надо выполнить

*atrm 1*

6. Добавьте в очередь команду *ps* с выводом в файл */tmp/ps.out*. Время выполнения – текущее время +5 минут.

7. Проверьте, что ваше задание находится в очереди.

8. После выполнения задачи из очереди посмотрите файл */tmp/ps.out*.

**Задание 7**. Изучите команду создания расписания процессов.

Ход выполнения:

1. Утилита *crontab* позволяет создавать и редактировать расписание повторяющихся задач. Для этого необходимо выполнить *crontab –e*. Спецификация задания имеет следующий формат:

*минуты часы день\_месяца месяц день\_недели задание*

Первые пять полей представляют собой шаблон времени и обязательно должны присутствовать. Чтобы программа игнорировала поле шаблона времени нужно поставить в нем символ звездочки (\*). Например:

*01 \* \* \* \* root /some/script*

Эта строка определяет, что сценарий /some/script должен выполняться каждую первую минуту каждого часа, каждого дня, каждого месяца и каждого дня недели. Сценарий будет выполняться от имени пользователя root.

2. Создайте расписание, в котором процессы запускаются:

* в первую минуту каждого часа;
* каждый день в 15:20;
* в 10 часов каждую субботу;
* в 9:30 каждое первое число месяца.

**Задание 8**. Изучите команды переключения режимы выполнения процесса.

Ход выполнения:

1. Для выполнения программы в режиме переднего плана достаточно набрать ее имя в командной строке и запустить на выполнение.

Для запуска программы в качестве фонового процесса следует набрать в командной строке ее имя и в конце добавить знак амперсанта (&), отделенный пробелом от имени программы, а также параметры командной строки, если таковые имеются. Затем программу запускают на выполнение. Вы получите примерно следующее сообщение (рис. 5):



*Рис. 5.* Пример запуска процесса в фоновом режиме.

Сообщение на рис. 5 состоит из двух чисел и приглашения командной строки. Это означает, что программа работает в фоновом режиме и есть возможность с этой же консоли запустить на выполнение еще какую-нибудь программу.

Число [1] означает номер запущенного фонового процесса, а 1849 – это идентификатор PID запущенного процесса.

2. Для проверки состояния фоновых процессов воспользуйтесь командой *jobs* (рис. 6).

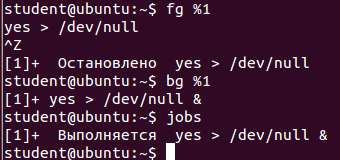


*Рис. 6.* Проверка состояния фонового процесса.

3. Для перевода процесса из фонового режима на передний план используйте команду fg %1, где 1 – номер фонового процесса.

4. Чтобы снова вернуть процесс в фоновый режим выполнения, его необходимо остановить, использую комбинацию клавиш Ctrl+Z. Затем можно перевести процесс в фоновый режим командой bg %1.

На рис. 7. отображены все эти действия.



*Рис. 7*. Переключение режимов выполнения процесса.

5. Запустите команду sleep 100 фоновом режиме. Проверьте ее состояние, а затем переведите ее на передний план. Что произойдет, если выполнить команду kill %<n>, где <n> – номер процесса в jobs?