|  |  |
| --- | --- |
| ДИСЦИПЛИНА | Технологии разработки серверных приложений |
| ИНСТИТУТ | Институт перспективных технологий и индустриального программирования |
| КАФЕДРА | Индустриального программирования |
| ВИД УЧЕБНОГО МАТЕРИАЛА | Практическая работа |
| ПРЕПОДАВАТЕЛЬ | Макиевский Станислав Евгеньевич |
| СЕМЕСТР | 4 семестр, 2024-2025 |

**Цель работы:** познакомиться с понятием процесса. Научиться получать список имеющихся в системе процессов и управлять их состоянием.

**Задание:**

**1. Изучение понятия процесса**

Операционная система Linux является многозадачной (мультизадачной). Это значит, что одновременно в системе может присутствовать множество процессов, каждому из которых доступно определенное количество процессорного времени. Для пользователя создается «иллюзия» одновременного выполнения процессов.

Процесс – выполняемая программа с ее данными и контекстом.

Каждый процесс имеет уникальный в любой момент времени номер в системе – PID. Первый запускающийся в системе процесс init, имеет PID = 1.

Для описания процессов в операционной системе имеется список структур – дескрипторов, содержащих информацию об идентификаторе процесса, приоритете, состоянии процесса, информацию о принадлежности пользователю и группе, занимаемых процессом ресурсах и др.

Каждый процесс в системе Linux запускается каким-либо процессом. Запускающий процесс – родительский, новый процесс – дочерний. Процессы, выполняющие одну задачу, объединяются в группы, имеющие собственный идентификатор. Процесс внутри группы, идентификатор которого совпадает с идентификатором группы процессов, считается лидером группы процессов.

Все запущенные процессы условно (в зависимости от выполняемой ими функции) можно разделить на три типа:

**Системные процессы** являются частью ядра и всегда расположены в оперативной памяти. Они часто не имеют соответствующих им программ в виде исполняемых файлов и всегда запускаются особым образом при загрузке ядра системы.

**Процессы-демоны** – это неинтерактивные процессы, которые выполняются в фоновом режиме.

К **прикладным** относятся все остальные процессы, выполняющиеся в системе.

Интерактивные процессы связаны с определенным терминалом и через него взаимодействуют с пользователем. Фоновые процессы выполняются независимо от пользователя и (псевдо)параллельно.

Каждый процесс в операционной системе Linux может находиться в одном или нескольких следующих состояний:

R (Runnable) – процесс или использует процессор или находится в очереди за получением процессорного ресурса;

S (Sleeping) – прерываемый сон;

D (Uninterruptable sleep) – непрерываемый сон;

T (Stopped) – процесс приостановлен (Ctrl-Z, SIGTSTP);

Z (Zombie) – процесс завершился, но код завершения не получен;

I (Idle kernel thread) – освобождение ресурсов потоков, которые были уничтожены;

< – повышенный приоритет;

+ – находится в интерактивном режиме.

1.1. Отобразите список процессов системы без и с использованием опций с помощью команды ps.

Основные опции:

–a – выдать все процессы системы, включая лидеров сеансов.

–d – выдать все процессы системы, исключая лидеров сеансов.

–e – выдать все процессы системы.

–x – выдать процессы системы, не имеющие контрольного терминала.

–o – определяет формат вывода в виде списка полей, разделенных символом «,».

–u – выдать процессы, принадлежащие указанному пользователю.

ps

ps -aux

ps -eux

Примечание:

PID, PPID – идентификатор процесса и его родителя.

%CPU – доля процессорного времени, выделенная процессу.

%MEM – процент используемой оперативной памяти.

VSZ – виртуальный размер процесса.

RSS – реальный размер процесса в памяти.

TTY – управляющий терминал.

STAT – статус процесса.

START – время запуска.

TIME – время исполнения на процессоре.

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

1.2. Получите выборочную информацию обо всех процессах в системе. Включите в формат: состояние, ID процесса, номер терминала и команду, которая запустила процесс.

ps -eo s,pid,tty,command

Примечание: поля, которые можно указывать после параметра -o, можно получить в

разделе «STANDARD FORMAT SPECIFIERS» справочной страницы по команде ps:

man ps | grep -A 400 '^STANDARD FORMAT' | less

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

1.3.Отобразите состояния процессов в реальном времени.

top

Утилита top в верхней части экрана показывает информацию по всей системе, в частности:

- общее время работы системы (uptime);

- среднюю загрузку системы в последние одну, пять и пятнадцать минут;

- информацию о количестве процессов и их состояниях;

- использование процессорного времени, оперативной памяти, области подкачки.

В нижней части экрана показывается информация о процессах. По умолчанию выводятся поля:

- PID;

- эффективный пользователь;

- приоритет;

- значение NICE;

- размер виртуальной памяти;

- количество физической памяти, занятой процессом;

- объем памяти, совместно используемой с другими процессами;

- состояние процесса;

- процент использования процессорного времени (после последнего обновления информации);

- процент использования физической памяти;

- общее количество использованного процессорного времени.

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

1.4. Получите справочную информацию по утилите top, находясь в интерактивном режиме работы этой утилиты (нажмите на клавишу h). Изучите действия, выполняемые по нажатию на клавиши u, f, r, d, L.

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

**2. Управление процессами**

Чтобы запустить программу достаточно ввести ее имя в командной строке и нажать «Enter». Однако не все команды запускают единственный процесс.

Интерактивные процессы, запущенные в терминале, занимают терминальную сессию, и оболочка не выводит пользователю строку приглашения до тех пор, пока программа не завершится.

Работу некоторых запущенных в терминале программ можно прервать с помощью сочетания клавиш «Ctrl + C» в окне терминала. В этот момент программе посылается сигнал INT (Interrupt).

Чтобы запустить программу в фоновом режиме необходимо завершить команду символом амперсанд «&». После этого в терминал выводится информация о запущенном процессе включая номер задания терминала, и приглашения пользователю на ввод новой команды.

Процессы разделения времени планируются, исходя из динамического приоритета.

Пользователь может влиять на динамический приоритет с помощью весового коэффициента – числа NICE.

Для процессов реального времени число NICE не имеет смысла.

Число NICE изменяется от -20 до 19. По умолчанию процессу назначается NICE равный 0.

Чем больше NICE, тем меньше динамический приоритет процесса.

Обычный пользователь может только увеличить NICE у процесса, уменьшить может только root.

Команда для запуска процесса с измененным NICE:

nice -n значение\_NICE командная\_строка

Изменить NICE у работающего процесса можно командой:

renice -n значение\_NICE -p PID\_процесса

2.1. Выведите список процессов пользователя root, используя команду ps. С помощью утилиты для подсчета количества строк посчитайте, сколько процессов принадлежит этому пользователю. Перенаправьте список процессов root в файл proc1.

ps -u root

ps -u root | wc -l

ps -u root > proc1

less proc1

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

2.2. Получите информацию о процессах вашего пользователя, имеющих статус работоспособный.

ps -u {пользователь} r

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

2.3. Добавьте к файлу proc1 сведения о процессе в данный момент потребляющий большой процент ресурсов центрального процессора.

top

ps -p {номер процесса, который потребляет большой процент ресурсов} >> proc1

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

2.4. Запустите браузер firefox в фоновом режиме в первом терминале. Во втором терминале запустите утилиту top в фоновом режиме.

firefox &

top &

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

2.5. В первом терминале найдите первый запущенный вами процесс с помощью команды ps и перенаправьте вывод этой команды на ввод команды grep. Определите ID процесса и номер терминала.

ps | grep firefox

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

2.6. Во втором терминале по идентификатору процесса найдите запущенный вами процесс, связанный с утилитой top.

ps | grep {ID\_процесса}

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

2.7. Вернитесь в первый терминал и найдите поддерево для процесса firefox, изучите список составляющих его процессов.

pstree

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

2.8. Измените приоритет для процесса, порожденного программой firefox. Сделайте его равным 10. Подтвердите, что приоритет изменился.

ps | grep firefox

renice -n 10 {номер процесса firefox}

ps -o s,pid,tty,command,nice | grep firefox | grep 10

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

Примечание:

У команды renice есть два полезных параметра:

-u имя\_учетной\_записи\_пользователя – изменить NICE у всех процессов пользователя

-g имя\_учетной\_записи\_группы – изменить NICE у всех процессов, содержащих указанный GID.

2.9. Воспользуйтесь командой sleep для ввода задержки на 200 секунд и получите список заданий текущей сессии терминала. Обратите внимание, что задание sleep завершится через 200 секунд.

sleep 200

jobs

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

Примечание: команда sleep используется в скриптах. Если вы используете команду sleep с аргументом X, то следующая команда может быть запущена только через X секунд. Команда имеет простой синтаксис: sleep Number[Suffix]

Здесь суффиксом может быть:

s – секунды (по умолчанию);

m – минуты;

h – час;

d – день.

2.10. Перенесите на передний план задачу sleep. Далее нажмите сочетание клавиш «Ctrl + C» и снова посмотрите список задач.

fg %2

jobs

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

2.11. Заново создайте задачу sleep и перенесите ее на передний план. Далее нажмите сочетание клавиш «Ctrl + Z» и снова посмотрите список задач. Обратите внимание на то, что процесс получил статус «Остановлен».

sleep 200

fg %2

jobs

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

2.12. Получите список сигналов для команды kill. Завершите запущенный процесс firefox двумя способами: с помощью сигналов SIGKILL и SIGTERM. Обратите внимание на разницу между статусами процесса.

kill -l

kill -9 {номер процесса firefox}

kill -15 {номер процесса firefox}

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

Примечание (самые часто используемые сигналы для процесса):

1 (SIGHUP) – при передаче службе происходит перечитывание конфигурационного файла службы, при выходе из bash рассылается всем порожденным процессам, завершая их.

2 (SIGINT) – аварийное завершение процесса (Ctrl-C).

3 (SIGQUIT) – дамп и аварийное завершение процесса (Ctrl-\).

9 (SIGKILL) – безусловное завершение процесса.

15 (SIGTERM) – завершение процесса, которое должен осуществить обработчик (перехватчик) сигнала.

SIGKILL вызывает немедленное завершение процесса. В отличие от SIGTERM или SIGINT этот сигнал не может быть перехвачен или проигнорирован, а процесс, получивший его, не имеет возможности выполнить какие-либо действия перед своим завершением.

2.13. Отправьте сигнал SIGKILL заданию sleep.

kill -9 %2

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

2.14. Завершите все процессы вашего пользователя с помощью команды killall.

killall -u {имя\_текущего\_пользователя}

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |

**3. Дополнительно**

3.1. Изучите информацию о доступе к общим библиотекам и просмотрите конфигурационные файлы.

Общие библиотеки – это скомпилированный код, предназначенный для совместного использования несколькими различными программами. Они распространяются в виде файлов .so в /usr/lib/.

Именования общих библиотек: имя.so.версия или имя-версия.so.

Команда ldd имя\_программа показывает, какие общие библиотеки требуются для запуска динамически скомпонованной программы.

Настройка местонахождения общих библиотек производится с помощью конфигурационного файла /etc/ld.so.conf или conf файлов в /etc/ld.so.conf.d.

В конфигурационные файлы помещаются имена каталогов, содержащих общие библиотеки.

Команда ldconfig создает кэш, в котором хранится информация о всех библиотеках (/etc/ld.so.cache).

Команда ldconfig -p выводит список библиотек, хранимый в кэше.

|  |  |
| --- | --- |
|  | student@prac-work-question:~#Предоставьте ответ в виде скриншота(-ов), где каждый шаг (действие) сопровождается письменным описанием.  Если необходимо предоставить скрипт, то ответ может содержать ссылку на скрипт решения (только git): |
|  | |
|  | |