**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

A logo of a triangle

Description automatically generated**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

|  |  |
| --- | --- |
| Институт | компьютерных наук |
| Кафедра | автоматизированных систем управления |

ЛАБОРАТОРНАЯ РАБОТА №3

По дисциплине "Операционные системы Linux"

На тему "Процессы и управление ими в операционной системе Linux"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ПИ-22-1 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Первеева Е. Ю. |
| Руководитель | |  |  |  |
| канд.техн.наук, доцент  ученая степень, ученое звание | |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Кургасов В.В. |

Липецк, 2024 г.

Оглавление

[**Цель работы** 3](#_Toc181193805)

[**Порядок выполнения работы** 4](#_Toc181193806)

[**1. Часть I** 4](#_Toc181193807)

[**2. Часть II** 10](#_Toc181193808)

[**3. Часть III** 14](#_Toc181193809)

[**4. Часть IV** 16](#_Toc181193810)

[**Контрольные вопросы** 23](#_Toc181193811)

[**Вывод** 29](#_Toc181193812)

**Цель работы**

Ознакомиться на практике с понятием процесса в операционной системе. Приобрести опыт и навыки управления процессами в операционной системе Linux.

**Порядок выполнения работы**

**1. Часть I**

1.1. Войти под пользовательской учетной записью (не root). Найти файл с образом ядра. Выяснить по имени файла номер версии Linux.

Файл с образом ядра в Linux находится в каталоге /boot и называется vmlinuz. Воспользуемся командой ls /boot | grep vmlinuz для выяснения версии Linux. Команда grep выполняет поиск по строке, содержащей vmlinuz. Она фильтрует вывод, показывая только строки, которые содержат эту последовательность символов. Результат выполнения команды представлен на рисунке 1.



Рисунок 1 - Файл с номером версии ядра Linux

vmlinuz-6.1.0.25-amd64 (это сжатый бинарный файл ядра), где 6.1.0.25 обозначают версию ядра.

1.2. Посмотреть процессы ps -f. Прокомментировать, изучив предварительно справку командой man ps.

Команда ps предоставляет список запущенных процессов в текущем сеансе терминала. -f опция позволяющая предоставлять расширенную информацию о процессах (без неё выводиться только PID, TTY, TIME, CMD).

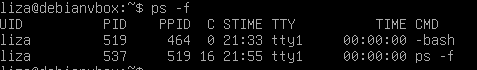


Рисунок 2 - Вывод активных процессов

Из рисунка 2 видно, что команда ps -f вывела два запущенных процесса: bash (запускается при входе в систему), сама команда ps -f.

UID - идентификатор пользователя, который запустил процесс

PID - уникальный идентификатор процесса

PPID - идентификатор родительского процесса (процесса, запустившего данный процесс)

C — процент процессорного времени, использованного процессом.

STIME — время старта процесса.

TTY — терминал, к которому процесс привязан (? указывает, что процесс не привязан к терминалу)

TIME — общее процессорное время, потраченное процессом.

CMD — команда, которая была выполнена для запуска процесса

1.3. Написать с помощью редактора vi два сценария loop и loop2.

С помощью редактора vi были созданы два файла loop и loop2 со следующими сценариями (Рисунок 3):

1) loop: while true; do true; done (бесконечный цикл без вывода в терминал)

2) loop2: while true; do true; echo ‘Hello’; done (бесконечный цикл, на каждой итерации на экран выводиться Hello)

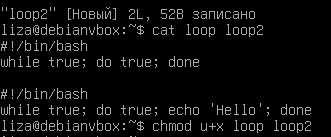


Рисунок 3 - Создание сценариев loop и loop2

1.4. Запустить loop2 на переднем плане.

Командой ./loop2 запустим сценарий loop2 на переднем плане (используем данную команду так как ранее разрешили исполнение этого файла) (рисунок 4).

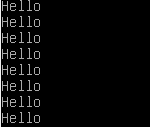


Рисунок 4 - Результат работы loop2

1.5. Остановить, послав сигнал STOP.

Сочетание клавиш Ctrl + Z приостановит процесс и отправит ему сигнал STOP, что временно остановит выполнение (Рисунок 5)

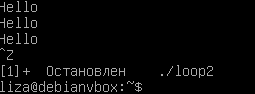


Рисунок 5 - Остановка процесса

1.6. Посмотреть последовательно несколько раз ps -f. Записать сообщение, объяснить.

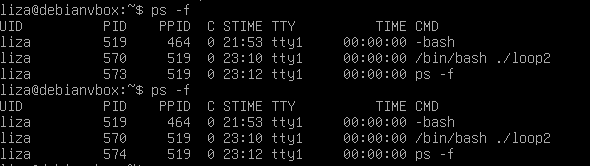


Рисунок 6 - Последовательный просмотр ps -f

На рисунке 6 видно, что TIME – затраченное время процессора на выполнение процесса не меняется после повторного запуска команды ps -f. Это свидетельствует о том, что процесс. /loop2 остановлен.

1.7. Убить процесс loop2, послав сигнал kill -9 PID. Записать сообщение. Прокомментировать.

Воспользуемся командой kill -9 570 чтобы убить процесс loop2 (Рисунок 7)

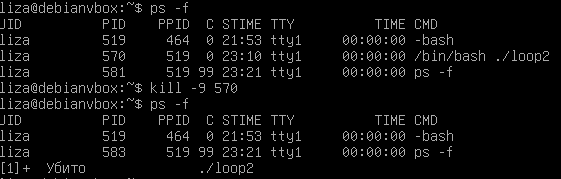


Рисунок 7 - Прекращения существования процесса

Из рисунка 7 видно, что процесс. /loop2, находившейся в остановке был убит и полностью утратил свое существование, что подтверждает команда ps -f.

Команда kill -9 PID используется для немедленного завершения процесса с заданным PID. Сигнал -9 представляет собой SIGKILL, который принудительно завершает процесс, не давая ему возможности сохранить данные или выполнить любые завершительные действия. Сигнал SIGKILL не может быть перехвачен или проигнорирован процессом, поэтому его использование может привести к сбоям.

1.8. Запустить в фоне процесс loop: sh loop &. Не останавливая, посмотреть несколько раз ps -f. Записать значение, объяснить.

Команда sh loop & запустит процесс loop в фоновом режиме (Рисунок 8).

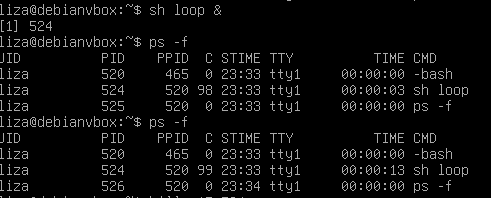


Рисунок 8 - Запуск процесса в фоне

На рисунке 8 видно, что после запуска процесса loop в фоновом режиме ему был автоматически присвоен PID 524 и номер задачи [1] (указывает на то, что процесс относится к фоновым процесса и управляется оболочкой). При первом запуске команды ps -f процесс loop занял 3 секунд процессорного времени, а при втором запуске13 секунд, это свидетельствует о том, что все это время процесс работал в фоновом режиме.

1.9. Завершить процесс loop командой kill -15 PID. Записать сообщение, прокомментировать.

Команда kill -15 PID используется для завершения процесса с сигналом SIGTERM, который запрашивает у процесса корректное завершение (Рисунок 9). В отличие от kill -9, SIGTERM позволяет процессу выполнить любые необходимые действия для завершения (например, сохранить данные).

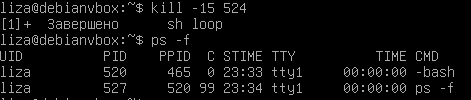


Рисунок 9 - Завершение процесса

1.10. Третий раз запустить в фоне. Не останавливая, убить командой kill -9 PID.

Запустим процесс loop в фоновом режиме, принудительно завершим командой kill -9 532, не останавливая сам процесс loop (Рисунок 10).



Рисунок 10 - Принудительное завершение процесса

1.11. Запустить еще один экземпляр оболочки: bash.

Для запуска еще одного TTY (терминала) необходимо воспользоваться сочетанием клавиш CTRL + ALT + F<номер оболочки>. На рисунке 12 представлен переход в TTY1 из TTY2.

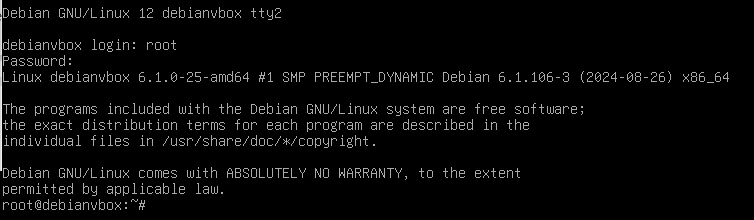


Рисунок 11 - Запуск ещё одного экземпляра оболочки

1.12. Запустить несколько процессов в фоне. Останавливать их и снова запускать. Записать результаты просмотра командой ps -f.

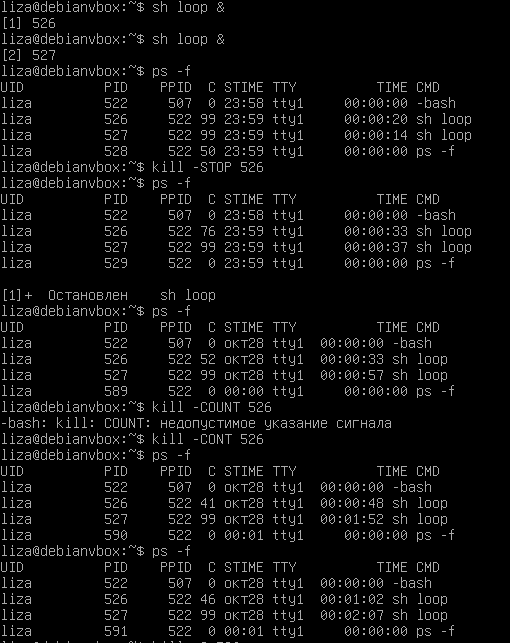


Рисунок 12 - Исследование фоновых процессов

**2. Часть II**

2.1. Запустить в консоли на выполнение три задачи: две в интерактивном режиме, одну – в фоновом.

Создадим файл loop3 с идентичным loop содержанием.

При запуске задачи в интерактивном режиме она занимает полностью терминал и не позволяет взаимодействовать с ним. Для решения этой проблемы запустим loop в фоновом режиме а loop2 в интерактивном в TTY1. После этого перейдем в терминал TTY2 (CTRL + ALT + F2) и запустим задачу loop3 в интерактивном режиме (Рисунок 13)

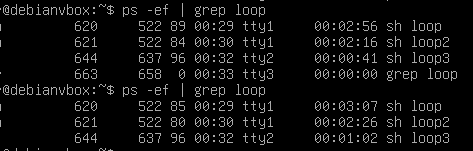


Рисунок 13 - Отслеживание процессов

Команда ps -ef | grep loop используется для поиска всех процессов, содержащих слово loop в своем имени или аргументах командной строки. -e показывает все процессы.-f выводит информацию в полном формате. | grep loop — фильтрует результат команды ps -ef, чтобы отобразить только строки, содержащие loop.

2.2. Перевести одну из задач, выполняющихся в интерактивном режиме, в фоновый режим.

Остановим процесс loop3 (CTRL + Z), возобновим в фоновом режиме командой bg %<job number> (рисунок 14).

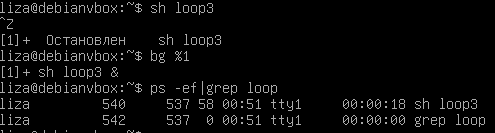


Рисунок 14 - Перевод процесса в фоновый режим

2.3. Провести эксперименты по переводу задач из фонового режима в интерактивный и наоборот.

Остановим процесс loop3 (kill -STOP PID), возобновим в интерактивном режиме командой fg %<job number> (рисунок 15).

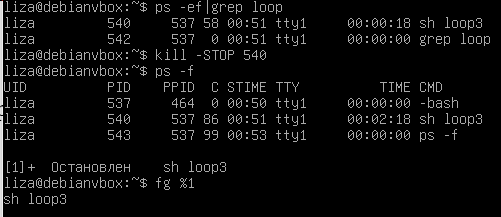


Рисунок 15 - Перевод процесса в интерактивный режим

Из рисунка 15 видно, что при переводе процесса loop3 в интерактивный режим он взял управление терминалом на себя и пользователь больше не может взаимодействовать с терминалом, пока не остановит процесс.

2.4. Создать именованный канал для архивирования и осуществить передачу в канал списка файлов домашнего каталога вместе с подкаталогами и одного каталога вместе с файлами и подкаталогами.

Именованный канал (FIFO) — это специальный тип файла в Unix-подобных системах, который позволяет организовать обмен данными между процессами. В отличие от обычного канала (pipe), именованный канал существует как файл в файловой системе и позволяет передавать данные между несвязанными процессами. Именованный канал создается с помощью команды mkfifo <name>. Командой ls -R вызываем вывод списка файлов домашней директории и ее подкаталогов и оператором ">" перенаправляем этот вывод в именованный канал (Рисунок 16).



Рисунок 16 - Создание именованного канала

Перейдём в TTY2 и при помощи команды cat и оператора перенаправления вывода "<" выведем содержимое kanal в терминал (Рисунок 17)



Рисунок 17 - Вывод содержимого канала

На рисунке 17 видно, что именованный канал не хранит данные, а предоставляет механизм для передачи их от одного процесса к другому, позволяя процессам взаимодействовать друг с другом.

Создадим директорию kanal-dir для архивации и воспользуемся командой tar -cvf kanal /home/liza/test-dir, которая архивирует данные из каталога kanal-dir и направит их в kanal (Рисунок 18)



Рисунок 18 - Передача архивных данных

Перейдем в другой терминал и попытаемся вывести содержимое kanal, как видно на рисунке 19 выводиться заархивированная, созданная ранее директория.



Рисунок 19 - Вывод содержимого kanal

Воспользуемся командой cat < kanal > test1.tar и с помощью файла test1.tar в который мы направили содержимое kanal выведем заархивированный директорий dir-test (Рисунок 20).

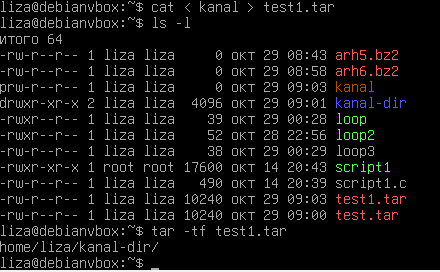


Рисунок 20 - направление содержимого kanal в файл

**3. Часть III**

Вариант 5

1. Отобразить информацию о процессах указанного пользователя в виде иерархии, вывод отсортировать по значениям PID.

Для отображения информации о процессах пользователя liza в виде иерархии и сортировки вывода по значениям PID можно использовать команду ps вместе с --forest (для отображения в виде дерева (иерархически)) и –sort (--sort=pid — сортирует вывод по значениям PID).

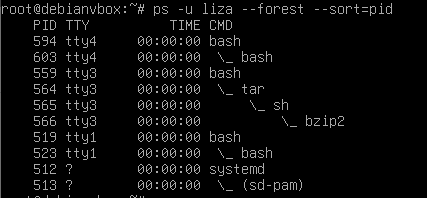


Рисунок 21 -Отображения информации о процессах пользователя liza

На рисунке 21 вывелось иерархическое дерево, в котором отображена информация о процессах указанного пользователя, отсортированная по PID.

2. С помощью сигнала SIGSTOP приостановить выполнение процесса, владельцем которого является текущий пользователь. Через несколько секунд возобновить выполнение процесса.

Приостановим процесс с помощью согнала SIGSTOP (остановит процесс, но он останется в памяти). Просмотрим результат остановки командой ps -f и по неизменившемуся времени работы процесса можно увидеть что он остановлен.

Затем возобновим выполнение процесса с помощью сигнала SIGCONT (возобновит выполнение процесса с того места, где он был остановлен) (Рисунок 22).

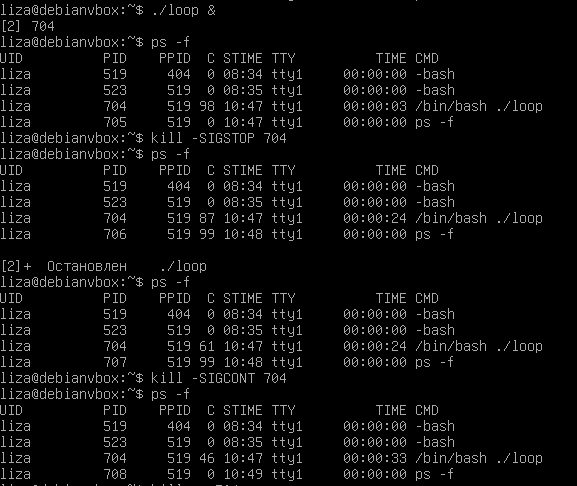


Рисунок 22 - приостановление и возобновление работы процесса

3. Определить идентификаторы и имена процессов, не связанных с указанным терминалом.

Воспользуемся командой ps -e -o pid,tty,comm | grep -v "tty1"| grep -v "?". ps -e — отображает все процессы в системе. -o pid,tty,comm — форматирует вывод, показывая только идентификаторы процессов (PID), терминал (TTY), и имя команды (comm). grep -v "tty1" — исключает строки, связанные с терминалом tty1, показывая только процессы без привязки к tty1. Чтобы не видеть процессы, не связанные с **никаким** терминалом добавим grep -v "?" (Рисунок 23).



Рисунок 23 - Идентификаторы и имена процессов, не связанных с указанным терминалом

На рисунке 23 отобразились процессы, их идентификаторы и терминалы, к которым они принадлежат, при этом были исключены процессы терминала 1 и процессы, вообще не принадлежащие терминалам.

**4. Часть IV**

4.1. Вывести общую информацию о системе

4.1.1. Вывести информацию о текущем интерпретаторе команд

Воспользуемся командой echo $SHELLчтобы получить путь к текущему интерпретатору команд, который хранится всегда в переменной SHELL (Рисунок 24)



Рисунок 24 - Путь к интерпретатору команд

4.1.2. Вывести информацию о текущем пользователе

Команда whoami выводит текущее имя пользователя, связанного с активным сеансом (Рисунок 25).



Рисунок 25 - Информация о пользователе

4.1.3. Вывести информацию о текущем каталоге

Команда pwd выводит информацию о текущем каталоге пользователя (Рисунок 26)



Рисунок 26 - Информация о директории

4.1.4. Вывести информацию об оперативной памяти и области подкачки

Команда free -h отобразит информацию об оперативной памяти и области подкачки (-h отображался размер данных) (Рисунок 27).

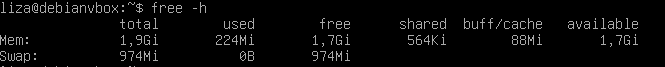


Рисунок 27 - Информация об оперативной памяти и области подкачки

Mem: оперативная память;

Swap: область подкачки;

shared: объем разделяемой памяти;

buff/cache: память, используемая для кеширования и буферов, которую можно освободить;

available: объем памяти, доступный для выделения новым процессам или уже существующим.

4.1.5. Вывести информацию о дисковой памяти

Команда df -h показывает информацию о файловых системах и использовании дискового пространства (-h также выводит размер данных) (Рисунок 28).



Рисунок 28 - Информация о дисковой памяти

4.2. Выполнить команды получения информации о процессах

4.2.1. Получить идентификатор текущего процесса

Выведем идентификатор текущего процесса с помощью команды echo $$, так как переменная $$ хранит идентификатор процесса оболочки, в котором выполняется команда (Рисунок 29)

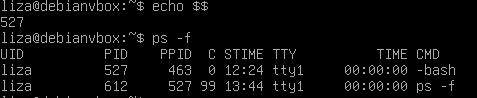


Рисунок 29 - идентификатор текущего процесса

4.2.2. Получить идентификатор родительского процесса

Выведем идентификатор родительского процесса с помощью команды echo $PPID, так как переменная PPID хранит идентификатор родительского процесса. Такой же идентификатор мы можем увидеть при выводе команды ps -f (Рисунок 30).

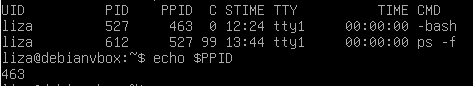


Рисунок 30 - Идентификатор родительского процесса

4.2.3. Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд

Воспользуемся командой ps -u $(whoami) (-u $(whoami) — фильтрует процессы, запущенные текущим пользователем. Команда $(whoami) динамически вставляет имя текущего пользователя) (Рисунок 31).

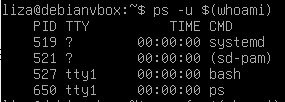


Рисунок 31 - Процессы текущего пользователя

4.2.4. Отобразить все процессы

Для отображения всех процессов воспользуемся командой ps -ef (f— выводит расширенную информацию обо всех процессах). Благодаря параметру -e выводятся все процессы, выполняющиеся в системе (Рисунок 32).

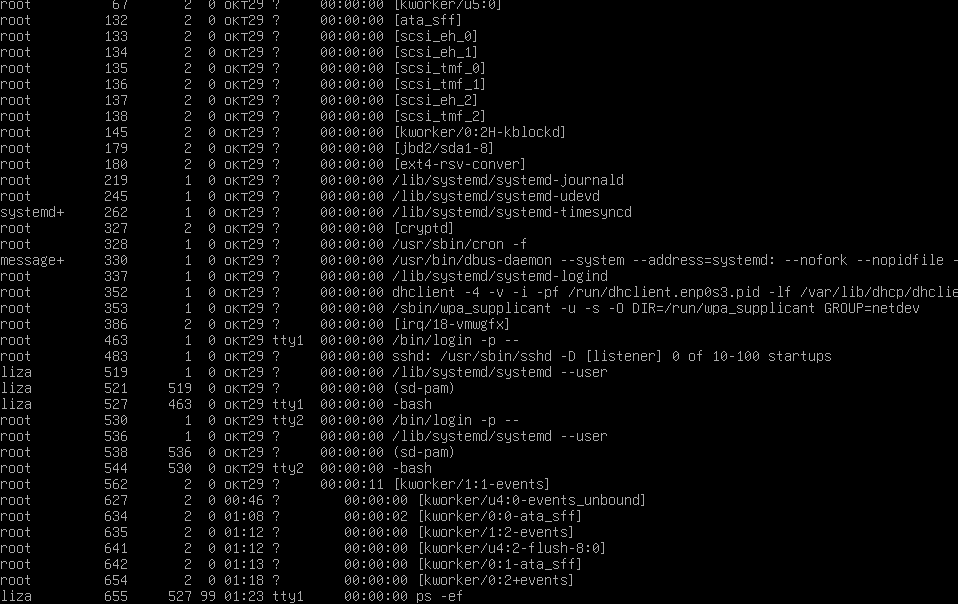


Рисунок 32 - Все процессы в системе

4.3. Выполнить команды управления процессами

4.3.1. Определить текущее значение nice по умолчанию

Команда nice используется для управления приоритетом процесса в Linux и UNIX-подобных системах. Она позволяет задавать приоритет процессу при его запуске, влияя на его доступ к ресурсам процессора.

Значение nice — это число, которое указывает "доброжелательность" процесса к другим процессам. Чем выше значение nice, тем меньше приоритета процесс будет иметь при выполнении (и наоборот). Значения nice могут варьироваться от -20 (самый высокий приоритет) до +19 (самый низкий).

По умолчанию текущее значение nice равно 0 (Рисунок 33).



Рисунок 33 - Приоритет nice по умолчанию

4.3.2. Запустить интерпретатор bash с понижением приоритета nice -n 10 bash

Команда nice -n 10 bash запускает новый экземпляр bash (оболочки) с приоритетом, установленным значением nice = 10. Это означает, что процессы, запущенные в этой оболочке, будут иметь пониженную приоритетность при распределении ресурсов процессора. Следовательно они будет получать меньше процессорного времени, чем процессы с более высоким приоритетом (Рисунок 34).

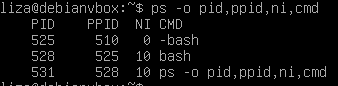


Рисунок 34 - Понижение приоритета

На рисунке 34 видно, что был запущен новый экземпляр bash с приоритетом (ni) равным 10.

4.3.3. Определить PID запущенного интерпретатора

На рисунке 34 видно, что PID запущенного интерпретатора равен 528, но чтобы в этом удостовериться выведем значение переменной $ (echo $$), которая хранит идентификатор процесса оболочки, в котором выполняются команды (Рисунок 35).



Рисунок 35 - PID запущенного интерпретатора

Как видно из рисунка 35 значение PID запущенного интерпретатора равно 528.

4.3.4 Установить приоритет запущенного интерпретатора равным 5

Чтобы повысить приоритет nice для уже запущенного процесса интерпретатора до 5, можно использовать команду renice и PID запущенного интерпретатора (Рисунок 36)

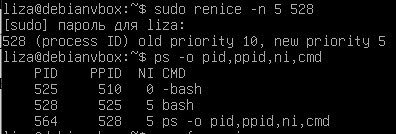


Рисунок 36 - Повышение приоритета запущенного интерпретатора

4.3.5. Получить информацию о процессах bash: ps lax | grep bash

Командой ps lax | grep sh получим информацию о процессах (Рисунок 37).

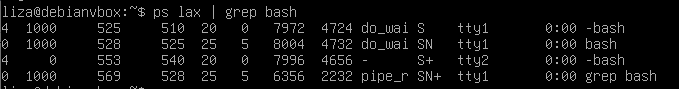


Рисунок 37 - Информация о процессах

**ps lax:**

l — формат вывода в виде длинного списка, включающий больше информации о процессе.

a — показывает процессы, запущенные всеми пользователями.

x — включает процессы, не привязанные к конкретному терминалу (например, фоновые или системные процессы).

**| grep bash:**

Команда grep bash фильтрует строки с именем bash, показывая только строки, где CMD равен bash (процессы оболочки bash).

**Вывод ps lax для каждого процесса включает следующие столбцы:**

F — флаги процесса (4 – использование прав администратора, 0 – отсутствие специальных флагов).

UID — идентификатор пользователя, которому принадлежит процесс.

PID — идентификатор процесса.

PPID — идентификатор родительского процесса.

PRI — текущий приоритет процесса.

NI — значение nice (приоритет планирования).

VSZ — виртуальный размер процесса в килобайтах.

RSS — объем памяти, который занимает процесс в физической памяти.

WCHAN — состояние, в котором находится процесс, ожидая события.

STAT — состояние процесса (R выполняющийся процесс; S спящий процесс; N указывает что процесс был запущен с пониженным приоритетом; присутствие + означает, что процесс является лидером группы процессов и управляется из текущей сессии в фоновом режиме.).

TTY — терминал, к которому прикреплен процесс.

TIME — время, которое процесс провел в состоянии выполнения.

CMD — имя команды, запускающей процесс.

**Контрольные вопросы**

1. Перечислите состояния задачи в ОС Linux.

R (Running) — Выполняется: Процесс активно выполняется или готов к выполнению, ожидая ресурсов процессора.

S (Sleeping) — Ожидание: Процесс приостановлен и ожидает выполнения какого-либо события (например, ввода-вывода). Это состояние может быть кратковременным.

D (Uninterruptible Sleep) — Непрерываемый сон: Процесс ожидает завершения операции, связанной с доступом к диску или другим ресурсам. В этом состоянии он не реагирует на сигналы, пока не завершится ожидаемое событие.

T (Stopped) — Остановлен: Процесс остановлен, обычно в результате действия сигналов SIGSTOP или SIGTSTP.

Z (Zombie) — Зомби: Процесс завершился, но его запись в таблице процессов пока не удалена, так как родительский процесс ещё не принял статус завершения этого процесса.

X (Dead) — Мёртвый: Процесс завершился и полностью удалён из системы. Этот статус встречается редко и используется в отладке.

2. Как создаются задачи в ОС Linux?

Основные способы создания задач включают:

Вызов fork() - Этот системный вызов создаёт новый процесс, называемый дочерним процессом, который является копией текущего процесса (родительского). После вызова fork() оба процесса продолжают выполнение, и fork() возвращает два разных значения: ноль в дочернем процессе и идентификатор дочернего процесса в родительском.

Команда system() - Эта функция позволяет выполнять команду оболочки в виде строки. Она фактически создаёт новый процесс, который выполняет указанную команду.

Команда shell - В командной строке или терминале можно запускать новые процессы, просто вводя эту команду (в том числе запускать и исполняемые файлы).

Задачи в фоновом режиме - При добавлении символа & в конце команды она запускается в фоновом режиме, позволяя пользователю продолжать использовать терминал.

Демоны - Некоторые процессы запускаются как демоны, создаваемые при старте системы или вручную. Обычно это выполняется с помощью systemd, init, или других менеджеров запуска.

3. Назовите классы потоков ОС Linux.

В ОС Linux выделяются три класса потоков для процессов:

- Потоки реального времени, обслуживаемые по алгоритмы FIFO (имеют наивысшие приоритеты и не могут вытесняться другими потоками, за исключением того же потока реального времени с более высоким приоритетом, перешедшего в состояние готовности. Это позволяет гарантировать, что важные задачи, требующие немедленного выполнения, будут выполнены без задержек.);

- Потоки реального времени, обслуживаемые в порядке циклической очереди (Находящийся в состоянии готовности поток выполняется в течение кванта времени, после чего поток помещается в конец своей очереди. Это позволяет избежать ситуации, когда один поток занимает все ресурсы системы, и обеспечивает более равномерное распределение времени выполнения между потоками.);

- Потоки разделения времени (Потоки этого класса имеют приоритеты в диапазоне от 100 до 139. Они управляются планировщиком, который выделяет им квант времени для выполнения, после чего они могут быть временно приостановлены для других потоков с более высоким приоритетом).

4. Как используется приоритет планирования при запуске задачи?

Приоритет планирования в Linux определяет, как процессы получают доступ к ресурсам системы, в частности к времени процессора. Каждый процесс имеет базовый приоритет, который может изменяться с помощью команд nice и renice.

При запуске процесса ему автоматически присваивается приоритет, и задачи с более высоким приоритетом (меньшими числовыми значениями) выполняются раньше, чем задачи с более низким приоритетом. В системе существуют два класса потоков: потоки реального времени, которые имеют наивысший приоритет и могут вытеснять другие потоки, и потоки разделения времени, которые используют алгоритм с временными квотами.

5. Объясните, что произойдет, если запустить программу в фоновом режиме без подавления потока вывода.

Если запустить программу в фоновом режиме без подавления потока вывода, она будет продолжать выводить свои данные в стандартный поток вывода (stdout) и стандартный поток ошибок (stderr) или запрашивать ввод пользователя (например, scanf или read).

6. Объясните разницу между действием сочетаний клавиш Ctrl^Z и Ctrl^C.

Ctrl + Z - Это сочетание клавиш отправляет сигнал SIGTSTP текущему фоновому процессу, что приводит к его приостановке (пауза).

Ctrl + C - Это сочетание клавиш отправляет сигнал SIGINT текущему процессу, что приводит к его немедленному завершению.

Таким образом, основное различие заключается в том, что Ctrl + Z приостанавливает процесс, в то время как Ctrl + C завершает его.

7. Опишите, что значит каждое поле вывода команды jobs.

Вывод команды jobs предоставляет информацию о текущих процессах в терминале, их статусе и идентификаторах.

[n]: это номер задания (job ID). Он используется для ссылки на конкретную задачу в командах, таких как fg, bg, и kill.

+ / -: этот символ указывает на "состояние" задания (+ указывает на последнее задание, которое было запущено в фоновом режиме или приостановлено, - указывает на предыдущее задание по сравнению с последним)

PID: это идентификатор процесса, который показывает уникальный номер, присвоенный процессу операционной системой.

Состояние: указывает текущее состояние задания (Running: Задание выполняется в данный момент; Stopped: Задание приостановлено; Done: Задание завершилось; Exited: Задание завершилось с ошибкой)

Команда: это команда или программа, которая была запущена.

8. Назовите главное отличие утилиты top от jobs.

top: отображает информацию о всех запущенных процессах в системе, включая использование ресурсов (CPU, памяти) и состояние процессов в реальном времени.

jobs: показывает только задачи, запущенные в текущем терминале, включая их состояние (приостановлены, выполняются в фоне) и идентификаторы заданий.

9. В чем отличие результата выполнения команд top и htop?

Top предустановлена во всех дистрибутивах Linux и не требует отдельной установки, htop же требует предварительной установки. Top не поддерживает прокрутку, htop позволяет прокручивать процессы по горизонтали и вертикали. Top использует простой текст и выделяет информацию жирным шрифтом, htop имеет цветной интерфейс и более наглядное отображение потребления ресурсов.

10. Какую комбинацию клавиш нужно использовать для принудительного завершения задания, запущенного в интерактивном режиме?

Для принудительного завершения задания, запущенного в интерактивном режиме, используется сочетание клавиш CTRL + C, которая посылает процессу сигнал SIGINT.

11. Какую комбинацию клавиш нужно использовать для приостановки задания, запущенного в интерактивном режиме?

Для приостановки задания в интерактивном режиме используется сочетание клавиш CTRL + Z, которое переводит процесс в состояние остановки, с помощью сигнала SIGTSTP.

12. Какая команда позволяет послать сигнал конкретному процессу?

Для отправки сигнала конкретному процессу используется команда kill, которая принимает PID процесса и тип сигнала в качестве аргументов.

13. Какая команда позволяет поменять поправку к приоритету уже запущенного процесса?

Команда renice используется для изменения приоритета уже запущенного процесса: renice -n <значение> <PID>.

14. Какая команда позволяет запустить задание с пониженным приоритетом?

Команда nice позволяет запустить задачу с пониженным приоритетом: nice -n 10 <команда>.

15. Какая команда позволяет запустить задание с защитой от прерывания при выходе из системы пользователя?

Для запуска задания с защитой от прерывания при выходе из системы пользователя используется команда nohup. Эта команда позволяет продолжать выполнение процесса даже после выхода из терминала или сессии: nohup команда &. Символ & в конце команды позволяет запустить процесс в фоновом режиме. Вывод программы будет перенаправлен в файл nohup.out по умолчанию, если не указано иное.

16. Какой процесс всегда присутствует в системе и является предком всех процессов?

Процесс с PID 1 – init, всегда присутствует в системе и является предком всех процессов. Он запускается первым при загрузке системы и порождает другие процессы.

17. Каким образом можно запустить задание в фоновом режиме?

Для запуска задания в фоновом режиме нужно добавить символ "&" в конце команды запуска.

18. Каким образом задание, запущенное в фоновом режиме, можно перевести в интерактивный режим?

Чтобы перевести задание, запущенное в фоновом режиме, в интерактивный режим, используется команда fg (foreground). Например, fg %1 вернет в интерактивный режим задание с номером 1, %1 — это номер задания, который можно узнать из вывода команды jobs.

19. Каким образом приостановленное задание можно перевести в интерактивный режим?

Если задание было приостановлено с помощью Ctrl+Z, оно будет в состоянии "Stopped", и команда fg вернет его к выполнению, переведя в интерактивный режим.

20. Что произойдет с заданием, выполняющимся в фоновом режиме, если оно попытается обратиться к терминалу?

Если задание в фоновом режиме пытается получить доступ к терминалу, оно получит сигнал SIGHUP (Hangup). Это происходит, когда программа пытается выполнить операции, требующие интерактивного взаимодействия с пользователем. После получения сигнала SIGHUP может произойти:

Задание может быть приостановлено: если задание не может получить доступ к терминалу, оно может быть приостановлено, и его состояние изменится на "Stopped".

В некоторых случаях, если программа пытается вывести информацию на экран, пользователю может быть показано сообщение об ошибке, указывающее на то, что терминал недоступен.

Задание может быть завершено: если программа не обрабатывает сигнал SIGHUP, она может завершиться.

21. Сколько терминалов может быть открыто в одной системе? Как перемещаться между терминалами (какие комбинации клавиш необходимо использовать)?

Количество виртуальных терминалов в системе обычно 6, но может быть увеличено в зависимости от конфигурации. Перемещаться между ними можно с помощью комбинаций клавиш CTRL + Alt + F<n>, где n – номер виртуального терминала (tty).

22. В чем отличие идентификаторов PID и PPID? При каких условиях возможна ситуация, когда PPID равен нулю или отсутствует?

PID (Process ID) — это уникальный идентификатор, который назначается каждому процессу в системе при его создании. Он позволяет различать процессы.

PPID (Parent Process ID) — это идентификатор родительского процесса, который создал данный процесс. Он указывает, к какому родительскому процессу принадлежит текущий процесс.

PPID позволяет отслеживать и управлять иерархией процессов, в то время как PID служит для уникальной идентификации каждого процесса.

Условия, при которых PPID может равняться нулю или отсутствовать:

PPID = 0: Когда процесс является системным процессом, который был запущен непосредственно ядром. Обычно это касается процессов, создаваемых во время загрузки системы.

Отсутствие PPID: В Linux всегда есть значение для PPID, но в случае его равенства 1, оно указывает на то, что родительский процесс больше не существует.

23. Поясните, от чего зависит максимальное значение PID?

Максимальное значение PID в системе зависит от настроек ядра и может быть изменено с помощью файла /proc/sys/kernel/pid\_max.

24. В каком случае, при создании нового процесса, его идентификатор (PID) будет меньше, чем у процесса, запущенного ранее?

Идентификатор процесса (PID) в Linux является уникальным целым числом, назначаемым каждому процессу. В нормальных условиях новый процесс получает следующий доступный PID, что обычно означает, что его PID будет больше, чем у ранее запущенных процессов. Однако существуют ситуации, когда новый процесс может иметь PID, меньший, чем у уже существующих процессов:

В случае, если все доступные PID-идентификаторы уже использованы, система начинает перераспределять PID с начала.

Также это возможно, если предыдущий процесс с меньшим PID завершился, освободив его для нового процесса.

**Вывод**

В ходе выполнения данной лабораторной работы ознакомился на практике с понятием процесса в операционной системе. Приобрел опыт и навыки управления процессами в операционной системе Linux.