

Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Автоматизированные системы обработки информации и управления»

Сёмкин П.С., Сёмкин А.П.

Методические материалы к лабораторным работам
по дисциплине
«Операционные системы»

Лабораторная работа № 9
«OS Alt Linux. Мониторинг и управление процессами»

Москва
2024 г.

ОГЛАВЛЕНИЕ

1	ЦЕЛЬ РАБОТЫ	4
2	ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	4
2.1	Подсистема управление процессами ОС Alt Linux.....	4
2.2	Процессы и потоки ОС Alt Linux	4
2.2.1	Понятие процесса ОС Linux.....	4
2.2.2	Создание процессов	4
2.2.3	Классы процессов Alt Linux.....	5
2.2.4	Граф состояния процесса	5
2.3	Планирование процессов в ОС Alt Linux	6
2.3.1	Назначение планирования процессов	6
2.3.2	Очередь выполнения.....	6
2.3.3	Статический приоритет процесса «nice»	7
2.3.4	Приоритет планирования процесса «priority».....	7
2.3.5	Политики планирования процессов	8
2.3.6	Планирование процессов реального времени.....	9
2.3.7	Кванты процессорного времени	9
2.3.8	Алгоритм диспетчеризации процессов	10
2.3.9	Многопроцессорное планирование.....	11
2.4	Управление процессами	11
2.4.1	Монитор процессов Htop.....	12
2.4.2	Монитор процессов «Системный монитор»	12
2.4.3	Утилиты командной строки для управления процессами	12
3	ВЫПОЛНЕНИЕ РАБОТЫ.....	13
3.1	Задание.....	13
3.2	Порядок выполнения работы	13
3.2.1	Выполнить мониторинг и управление процессами с помощью монитора процессов htop.....	14
3.2.2	Выполнить мониторинг и управление процессами с помощью графической утилиты «Системный монитор»	14
3.2.3	Выполнить мониторинг и управление процессами с помощью утилит командной строки	15
4	КОНТРОЛЬНЫЕ ВОПРОСЫ.....	15
5	ЛИТЕРАТУРА	16
6	ПРИЛОЖЕНИЕ.	17
6.1	Монитор процессов htop.....	17
6.1.1	Запуск монитора	17
6.1.2	Область с общей информацией о системе	17

6.1.3	Область с подробной информацией о процессах.....	17
6.1.4	Назначение функциональных клавиш.....	18
6.1.5	Клавиши для перемещения по окну программы.....	19
6.1.6	Поиск и фильтрация процессов.....	19
6.1.7	Изменение приоритета процесса.....	19
6.1.8	Выбор нескольких процессов.....	20
6.1.9	Закрытие процесса.....	20
6.1.10	Отображение только процессов определённого пользователя.....	20
6.1.11	Отображение только процессов с определёнными номерами.....	20
6.2	Утилита «Системный монитор».....	20
6.2.1	Вкладка «Система».....	20
6.2.2	Вкладка «Процессы».....	21
6.2.3	Вкладка «Ресурсы».....	21
6.2.4	Вкладка «Файловые системы».....	21
6.3	Команды интерпретатора BASH для мониторинга и управления процессами.....	22
6.3.1	Команды получения общей информации о системе.....	22
6.3.2	Команды получения информации о процессах.....	22
6.3.3	Команды управления процессами.....	23

1 Цель работы

Целью работы является знакомство и практическая работа со средствами мониторинга и управления процессами ОС Alt Linux.

2 Теоретическая часть

2.1 Подсистема управление процессами ОС Alt Linux

Подсистема управления процессами предназначена для обеспечения эффективной реализации многозадачного режима в ОС Alt Linux.

Подсистема отвечает за **распределение процессоров** для выполнения процессов.

В состав подсистемы управления процессами входит **планировщик процессов (process scheduler)**, обеспечивающий выделение процессорного времени процессам.

2.2 Процессы и потоки ОС Alt Linux

2.2.1 Понятие процесса ОС Linux

Процесс(**process**) ОС Linux - единица управления и единица потребления ресурсов. Процесс имеет один или несколько потоков выполнения(**thread**)

В системе процессы и потоки представляют собой единую структуру данных и носят название задачи(**task**).

Каждый процесс имеет собственное виртуальном адресном пространство, отображаемое в образ процесса в виртуальной памяти операционной системы

2.2.2 Создание процессов

Системный вызов **fork**, выполняемый родительским процессом (**parent process**) создаёт новый процесс, называемый дочерним процессом (**child process**), образ которого создаётся по принципу копирования образа родителя (**copy-on-write**). Образ потомка перезаписывается системным вызовом **execv**.

Системный вызов **vfork** позволяет создавать процесс без копирования страниц родительского процесса

Системный вызов **clone** создает новый поток либо в текущем процессе, либо в новом процессе.

Если новый поток находится в текущем процессе, он совместно использует с остальными потоками адресное пространство процесса

Если поток создаётся в новом процессе, то используется та же семантика, что и у системного вызова **fork**

2.2.3 Классы процессов Alt Linux

1. **Процессы реального времени (real_time), обслуживаемые по алгоритму FIFO**, имеют наивысшие приоритеты и не могут прерываться другими процессами, за исключением такого же процесса реального времени FIFO, перешедшего в состояние готовности.

2. **Процессы реального времени (real_time), обслуживаемые в порядке циклической очереди**, представляют собой то же самое, что и процессы реального времени FIFO, но с тем отличием, что они могут прерываться по таймеру.

3. **Процессы разделения времени (sharing_time)** обслуживаются в режиме пакетной обработки с выделением им заданных квантов времени.

2.2.4 Граф состояния процесса

Процесс переходит в состояние **выполнение (running)** после выделения ему процессорного времени.

При блокировке процесс переходит в состояние **спячка (sleeping)**, а при остановке работы в состояние **останов (stopped)**.

Состояние **зомби (zombie)** показывает, что выполнение процесса прекратилось, однако он еще не был удален из системы.

Процесс в состоянии **dead (смерти)** может быть удален из системы.

Состояния **active (активный)** и **expired (неактивный)** используются при планировании выполнения процесса.

При загрузке ядра запускается первый процесс **init**, который использует ядро для создания всех остальных процессов.

2.3 Планирование процессов в ОС Alt Linux

2.3.1 Назначение планирования процессов

Назначением планирования является:

- **максимизация пропускной способности системы** - числа процессов, выполняемых за единицу времени;
- **минимизация времени ожидания** - времени, прошедшего с момента готовности процесса до начала его выполнения;
- **минимизация времени ответа** - времени, прошедшего с момента готовности процесса до завершения его выполнения;
- **максимизация оптимального распределения ресурсов** между процессами.

2.3.2 Очередь выполнения

После создания процесса, он помещается в **очередь выполнения** (run queue) планировщика, содержащую ссылки на все процессы, требующие процессорное время.

Очереди выполнения представляют собой многоуровневые очереди с обратной связью, что позволяет присваивать процессам различные приоритеты для использования процессорного времени.

Когда процесс переходит в состояние блокировки либо спячки (т.е. ожидания), или же ее выполнение прекращается по какой-либо иной причине, процесс удаляется из очереди выполнения.

Чтобы отличить процессы, обладающие правом на процессорное время, от задач, которые вынуждены ожидать до наступления следующего периода дискретизации, в планировщике определены два состояния процесса: **активный (active)** и **неактивный (expired)**.

При этом планировщик осуществляет диспетчеризацию только тех процессов, которые находятся в активном состоянии.

Одной из целей планировщика Alt Linux является обеспечение высокой степени интерактивности системы.

2.3.3 Статический приоритет процесса «nice»

Каждому процессу присваивается статический приоритет (**static priority**), называемый также правильным значением (**nice value**) или «значением **nice**».

Значение **nice** устанавливается для каждого процесса в момент порождения этого процесса и при обычном запуске команд или программ принимается равным приоритету родительского процесса. «Значение **nice**» лежит в пределах от **-20** (самый высокий приоритет), до **+19** (самый низкий приоритет).

Системный вызов **nice** позволяет изменять значение **nice** при запуске программы.

Отрицательные значения nice может устанавливать только суперпользователь.

Остальные пользователи могут применять эту команду только для запуска процесса с низким значением приоритета.

Системный вызов **renice** предназначен для изменения значения **nice** для уже выполняющихся процессов.

Суперпользователь может изменить приоритет **любого процесса** в системе.

Остальные пользователи могут изменять значение приоритета только для тех процессов, для которых данный пользователь является **владельцем**. При этом обычный пользователь может только **уменьшить** значение приоритета (увеличить значение **nice**), но не может увеличить приоритет (даже для возврата значения **nice** к значению, устанавливаемому по умолчанию). Поэтому процессы с низким приоритетом не могут породить "высокоприоритетных потомков".

2.3.4 Приоритет планирования процесса «priority»

У каждого процесса есть **приоритет планирования (priority)**. Значение приоритета планирования **Alt Linux** может изменяться в промежутке от **0** до **39**.

По умолчанию приоритет планирования равен **20**.

Меньшему значению приоритета планирования соответствует более высокий приоритет процесса.

В планировщике Alt Linux приоритет планирования процесса влияет на размер кванта процессорного времени, выделяемого данному процессу, и порядок выполнения процессов.

Приоритет планирования равен значению статического приоритета **nice** задачи **плюс двадцать**.

2.3.5 Политики планирования процессов

Алгоритмы планирования процессов Alt Linux:

1. **SCHED_OTHER. min/max priority 0/0.** Алгоритм планирования на основе деления времени. Порядок выполнения определяется значением статического приоритета **nice**.

2. **SCHED_FIFO. min/max priority 1/99.** Простой алгоритм планирования по принципу «первым пришел – первым обслужен» без использования квантов времени. Предназначен для **класса процессов реального времени, обслуживаемых по алгоритму FIFO.**

3. **SCHED_RR. min/max priority 1/99.** Алгоритм планирования с квантованием. Циклический (round - robin) алгоритм. Предназначен для **класса процессов реального времени, обслуживаемых в порядке циклической очереди.**

4. **SCHED_BATCH. min/max priority 0/0.** Алгоритм планирования на основе деления времени с учётом значения приоритета **nice**. Приоритет процессов ниже, чем у процессов, планируемых на основе **SCHED_OTHER**.

5. **SCHED_IDLE. min/max priority 0/0.** Процессам, использующим такую политику, присваивается самый низкий приоритет.

6. **SCHED_DEADLINE. min/max priority 0/0.** Алгоритм планирования реального времени по ближайшему сроку выполнения(EDF))

2.3.6 Планирование процессов реального времени

Планировщик поддерживает жесткое планирование реального времени, пытаясь минимизировать время, затрачиваемое процессом реального времени на ожидание процессора.

В отличие от **обычного процесса**, который после окончания кванта времени помещается в **список неактивных**, **процесс реального времени** всегда помещается в **список активных процессов** по истечению выделенного ему кванта времени.

Процессы реального времени всегда выполняются с более высоким приоритетом, чем обычные процессы.

Поскольку планировщик всегда выделяет процессор процессу с самым высоким приоритетом из списка активных процессов (а процессы реального времени всегда находятся в активном списке), обычный процесс никогда не сможет вытеснить процесс реального времени.

Чтобы предотвратить случайное, либо злонамеренное, использование процессов реального времени, право на их создание имеют только пользователи с привилегиями **root**.

2.3.7 Кванты процессорного времени

С каждым процессом связан **квант процессорного времени**, то есть количество тиков таймера, в течение которых процесс может выполняться. По умолчанию системные часы тикают с частотой **100 Гц**, так что каждый тик равен **10 мс**. Этот интервал в ОС Alt Linux называют «**джиффи**» (jiffy - мгновение, миг, момент).

Квант времени – числовое значение, определяющее, как долго может выполняться процесс до того момента, пока он не будет вытеснен с процессора.

Слишком большое значение кванта времени приводит к ухудшению интерактивной производительности системы.

Слишком малое значение кванта времени приводит к возрастанию накладных расходов на переключение между процессами.

В планировщике Linux процессам выделяется доля (portion) процессорного времени. На величину доли влияет загруженность системы и значение параметра nice

2.3.8 Алгоритм диспетчеризации процессов

Планировщик процессов выполняет диспетчеризацию процессов следующим образом:

1. Планировщик вычисляет называемую в ОС Linux «добродетелью» (goodness) величину каждого активного процесса по следующему алгоритму:

if (class = real_time) goodness = 1000 + priority;

if (class = timesharing & quantum > 0) goodness = quantum + priority;

if (class = timesharing && quantum = 0) goodness = 0;

Для обоих классов реального времени выполняется первое условие.

Все, что дает пометка процесса, как процесса реального времени, это гарантия, что этот процесс получит более высокое значение **goodness**, чем все процессы разделения времени.

Если у процесса, который запускался последним, осталось неиспользованное процессорное время, он получает бонус, позволяющий выиграть в спорных ситуациях. Идея состоит в том, что при прочих равных условиях более эффективным представляется запустить предыдущий процесс, так как его страницы и кэш с большой вероятностью еще находятся на своих местах.

2. Когда происходит переключение процессов, выбирается процесс с максимальным значением «добродетели».

3. Во время выполнения процесса его квант (переменная **quantum**) уменьшается на единицу на каждом тике.

4. Процесс снимается с выполнения процессором при выполнении одного из следующих условий:

- Квант процесса уменьшился до 0.

- Процесс прерывается (блокируется на операции ввода-вывода, семафоре и т. д.)

- **В состоянии готовности перешёл ранее заблокированный процесс с более высокой «добродетелью».**

5. Каждый раз при удалении процесса из процессора, планировщик вычисляет для него следующий квант времени.

6. В случае блокирования процесса, либо невозможности его выполнения по иной причине, он деактивируется (**deactivate**), то есть удаляется из очереди выполнения до тех пор, пока не будет снова готов к выполнению.

2.3.9 Многопроцессорное планирование

Поскольку планировщик процессов управляет процессами с помощью отдельных для каждого процессора очередей выполнения, процессы, как правило, являются структурно связанными с определенным процессором.

Это означает высокую вероятность отправки процесса в последующих периодах дискретизации на тот же самый процессор, что повышает быстродействие процесса, если его данные и инструкции все еще находятся в процессорном кэше.

Однако такая схема может привести к простоям одного или нескольких процессоров в многопроцессорной системе даже в то время, когда система испытывает большую нагрузку. Во избежание подобной ситуации в случае обнаружения простоя процесса, планировщик осуществляет **балансировку загрузки (load balancing)** для переноса процесса с одного процессора на другой с целью повышения эффективности использования ресурсов.

Определение загрузки процессора выполняется планировщиком на основе данных о средней длине каждой очереди выполнения в течение нескольких последних прерываний таймера, чтобы минимизировать эффект непостоянства процессорной загрузки в алгоритме балансировки.

2.4 Управление процессами

Управление процессами ОС Alt Linux включает в себя выполнение следующих задач:

- *просмотр запущенных процессов*

- *просмотр информации о процессах*
- *поиск процессов*
- *изменение приоритета процессов*
- *завершение процессов*
- *ограничение памяти доступной процессу*

В Alt Linux есть множество утилит, как утилит командной строки, так и графических утилит, для решения различных задач по управлению процес-
сами.

2.4.1 Монитор процессов Htop

Монитор предназначен для просмотра в реальном времени информации о запущенных в системе процессах. Может быть запущен как из командной строки, так и из графической оболочки.

Htop показывает динамический список системных процессов, обычно выравненный по использованию ЦПУ. Показывается время непрерывной ра-
боты, использование процессоров и памяти.

Интерфейс программы приведён в приложении.

2.4.2 Монитор процессов «Системный монитор»

Системный монитор относится к графическим утилитам и позволяет просматривать список запущенных процессов, завершать процессы, следить за использованием памяти, центрального процессора и файловых систем.

Интерфейс утилиты приведён в приложении.

2.4.3 Утилиты командной строки для управления процессами

ОС Alt Linux, как и все ОС Linux, содержит большой набор утилит ко-
мандной строки для управления процессами

1. Утилита **ps**. Предназначена для мониторинга процессов в режиме реального времени. Показывает список всех процессов, которые выполнялись на момент запуска утилиты. Используется во многих случаях для вывода ин-
формации о выполняемых процессах. Может выполняться как в режиме ко-

мандной строки, так и имеет графическую оболочку. Команда содержит множество параметров, наиболее часто используемые из которых приведены в приложении

2. Утилита **top**. Предназначена для вывода информации о процессах в реальном времени. Процессы сортируются по максимальному занимаемому процессорному времени, но есть возможность изменить порядок сортировки. Утилита также выводит информацию о свободных системных ресурсах.

3. Утилита **nice**. Позволяет определить текущего значения приоритета **nice** по умолчанию, понижать приоритет запускаемого или выполняемого процесса. Пользователь с правами **root** может повышать приоритет команды.

4. Утилита **kill**. Предназначена для принудительного завершения процесса.

5. Утилиты **jobs**, **fg**, **bg** предназначены для перевода выполнения процессов в фоновый режим или режим переднего плана.

Синтаксис утилит управления процессами приведён в приложении.

3 Выполнение работы

3.1 Задание

Познакомиться с возможностями утилит Alt Linux для решения задач мониторинга и управления процессами операционной системы.

Выполнить команды управления процессами и объяснить полученные результаты.

3.2 Порядок выполнения работы

1. Войти в систему под учётной записью **stud_XX** (XX –индекс группы).
2. Запустить программу **Oracle VM VirtualBox**
3. Запустить виртуальную машину **Alt-10**
4. Войти в систему под учётной записью **admin_kaf**.

3.2.1 Выполнить мониторинг и управление процессами с помощью монитора процессов htop

1. Запустить программу **VirtualBox**
2. Определить **PID** программы **VirtualBox**
3. Запустить монитор **htop** для отображения информации о процессе **VirtualBox**
4. Проанализировать основные параметры процесса **VirtualBox** и всех дочерних процессов (общий объём виртуальной памяти, объём резидентной памяти, объём разделяемой памяти (общих страниц))
5. Определить приоритеты **планирования** и значение **nice**, присвоенные процессу **VirtualBox** и дочерним процессам при запуске программы
6. Повысить приоритет **nice** процесса **VirtualBox** и всех дочерних процессов до максимального
7. Завершить выполнение монитора

3.2.2 Выполнить мониторинг и управление процессами с помощью графической утилиты «Системный монитор»

1. Запустить программу «blender»
2. Запустить программу «Системный монитор»
3. Перейти на вкладку «Система» и проанализировать полученную информацию
4. Перейти на вкладку «Процессы»
5. Выделить процесс «blender»
6. С помощью контекстного меню просмотреть информацию о процессе и изменить приоритет процесса до **Высокий**
7. Используя вкладку **Ресурсы** отобразить и проанализировать информацию о системных ресурсах
8. Используя вкладку **Файловые системы** отобразить информацию об устройствах и файловых системах
9. Объяснить полученные результаты

3.2.3 Выполнить мониторинг и управление процессами с помощью утилит командной строки

1. Открыть окна интерпретатора команд
2. Получить общую информацию о системе
 - вывести информацию о текущем интерпретаторе команд
 - вывести информацию о текущем пользователе
 - вывести информацию о текущем каталоге
 - вывести информацию об оперативной памяти и области подкачки
 - вывести информацию о файловой системе
3. Получить информации о процессах
 - получить идентификатор текущего процесса(**PID**)
 - получить идентификатор родительского процесса(**PPID**)
 - получить идентификатор процесса по его имени(**init**)
 - получить информацию о выполняющихся процессах с помощью команды **ps**
1. Выполнить команды управления процессами
 - получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе
 - определить текущее значение **nice** по умолчанию
 - определить **PID** запущенного командного процессора
 - установить значение **nice** командного процессора равным **5**

4 Контрольные вопросы

1. Как создаются задачи в ОС Alt Linux?
2. Назовите состояния процесса в ОС Alt Linux.
3. Назовите классы процессов ОС Alt Linux.
4. Как устанавливается значение nice при запуске процесса?
5. Как можно изменить приоритет для выполняющегося процесса?

5 Литература

1. Робачевский А.М. Операционная система UNIX.-СПб.: БХВ-Петербург, 2001. – 528 с.:ил.
2. Негус К. Ubuntu и Debian Linux для продвинутых. 2-е изд. – СПб.: Питер,2014. -384 с.: ил.

6 Приложение.

6.1 Монитор процессов htop

6.1.1 Запуск монитора

\$ sudo htop <параметры>

После запуска открывается окно программы, разделённое на две части:

- **общая информация о системе**
- **подробная информация о процессах**

6.1.2 Область с общей информацией о системе

CPU - нагрузка на каждое ядро центрального процессора (цифры от 1 до 12).

Mem - общее количество оперативной памяти и используемая память.

Swp – статистика файла подкачки (если он есть)

Task - статистика по процессам

Load average - средняя загрузка центрального процессора

Uptime - время работы операционной системы с момента последней загрузки

6.1.3 Область с подробной информацией о процессах.

Значение столбцов:

PID - идентификатор процесса.

USER - имя пользователя-владельца процесса или ID если имя не может быть определено.

PRI - приоритет планирования - внутренний приоритет ядра для процесса.

Значение по умолчанию равно **20**. Равен значению **nice** процесса **плюс двадцать**.

Приоритет планирования меняется от **0** до **39**.

Чем меньше приоритет планирования, тем больше времени отводится процессу (отличается для процессов, имеющих приоритет выполнения **real-time**).

NI - значение **nice** процесса. Изменяется от **19** (низкий приоритет) до **-20** (высокий приоритет). Может быть изменено клавишами **F7** и **F8**.

VIRT – общий объём виртуальной памяти процесса (**M_SIZE**). Включает в себя: область кода (**CODE**), данные (**DATA**), разделяемые библиотеки (**SHARED**) и страницы, перемещенные в swar-область памяти (**CODE** - объём памяти, содержащий исполняемый код процесса. **DATA**- объём памяти, занятой данными, используемыми процессом в ходе выполнения. **SWAP** - объём памяти, используемой процессом, но перемещенной в swar-область.)

RES - объём резидентной (не перемещаемой в swar) памяти (text + data + stack) процесса (размер используемой физической памяти процесса, **M_RESIDENT**).

SHR – объём общих страниц процесса (**M_SHARE**) т.е. памяти, которая может быть использована другими приложениями.

S (STATE) - состояние процесса:

S - для спящих (в простое);

R - для запущенных (в состоянии выполнения);

D – для ожидающих;

Z для зомби;

CPU% - процент процессорного времени, которое процесс использует в данный момент.

MEM% - процент памяти, используемой процессом в данный момент (в зависимости от размера резидентной памяти процесса (**M_RESIDENT**)).

TIME+ - время, измеренное в часах, указывает на то, сколько процесс провёл в пользовательском и системном режимах.

Command – полный путь к программе процесса (имя программы и аргументы).

6.1.4 Назначение функциональных клавиш

F1 - справка;

F2 - настройки;

F3 - поиск процесса;

F4 - сортировка списка процессов;

F5 - вывод процессов в виде дерева;

F6 - выбор параметра для сортировки;

F7(Nice -) - увеличить приоритет **nice** выделенного процесса;

F8(Nice +) - уменьшить приоритет **nice** выделенного процесса;

F9 – убить процесс;

F10 – выйти из программы.

6.1.5 Клавиши для перемещения по окну программы

Клавиши **↑ ↓** - для прокрутки списка процессов.

Клавиши **PgUp PgDn** - для прокрутки окна.

Кнопка **Home** - перемещение к началу списка.

Кнопка **End** – перемещение к концу списка.

Ctrl-a - прокрутка к началу строки.

Ctrl-e - прокрутка к концу строки.

6.1.6 Поиск и фильтрация процессов

Поиск отличается от фильтрации тем, что найденные процессы показываются наравне с остальными, и между найденными процессами можно переключаться кнопкой **F3**.

При фильтрации на экран будут выводиться только процессы, соответствующие введённой строке.

Для перехода к поиску по процессам надо нажать **F3**. Для переключения между найденными процессами нажать **F3**.

Для фильтрации процессов нажать **F4** и ввести имя процесса.

Для очистки фильтра вновь нажать **F4** и затем Esc.

6.1.7 Изменение приоритета процесса

Для увеличения приоритета процесса (вычитание из величины **nice**) необходимо нажать **F7**. Эту операцию может выполнять только суперпользователь (пользователь **root** или если **htop** запущена с использованием утилиты sudo).

Операционные системы Лаб. работа № 9 (ОС Alt Linux. Мониторинг и управление про- 20
цессами)

Для уменьшения приоритета процесса (прибавления к величине nice)
надо нажать **F8**.

6.1.8 Выбор нескольких процессов

Для выбора процессов необходимо использовать **Пробел**. После этого
введённые команды, такие как **kill** или **изменение приоритета**, могут приме-
няться к группе выделенных процессов вместо подсвеченного в данный мо-
мент. Для снятия выделения со всех процессов нажать **U (Shift+u)**.

6.1.9 Закрытие процесса

Для закрытия процесса надо выбрать один или несколько процессов и
нажать **F9** или. Выбранному процессу будет отправлен сигнал завершения.
Если не отмечен ни один процесс, то будет закрыть тот, на котором находится
в данный момент курсор.

6.1.10 Отображение только процессов определённого поль- зователя

```
$ sudo htop -u <имя пользователя>
```

6.1.11 Отображение только процессов с определёнными но- мерами

```
$ sudo htop -p <PID,PID....>
```

6.2 Утилита «Системный монитор»

Для запуска Системного монитора следует перейти
Меню > Приложения > Системные > Системный монитор MATE.

Вся информация в окне программы распределена по четырем вкладкам:

6.2.1 Вкладка «Система»

Выводится основная информация об установленной операционной си-
стеме и оборудовании:

- имя компьютера
- версия ОС
- версия ядра Linux
- объём памяти

- информация о процессоре
- информация о доступном дисковом пространстве

6.2.2 Вкладка «Процессы»

Вкладка позволяет просматривать и управлять запущенными процессами. Каждый процесс можно **приостановить, остановить, изменить приоритет** и выполнить некоторые другие действия;

При щелчке правой кнопкой мыши по любому запущенному процессу, открывается контекстное меню, с помощью которого можно завершить «зависшее» приложение, остановить, перезапустить и изменить его приоритет, что позволяет регулировать требования процесса к системным ресурсам.

Для изменения приоритета процесса необходимо:

1. Выбрать вкладку **Процессы**, чтобы отобразить список процессов;
2. Выбрать процесс, приоритет которого следует изменить.
3. В контекстном меню процесса выбрать пункт **Изменить приоритет**.
4. Если выбрать пункт **Вручную**, откроется диалоговое **окно Изменить приоритет процесса**.
5. Можно использовать ползунок, чтобы установить уровень приоритета. Приоритет процесса задается уровнем **nice**. Меньшее значение **nice** соответствует более высокому приоритету.
6. Нажать кнопку **Изменить приоритет**.

Для установки более высокого приоритета, чем тот, который уже установлен у процесса, потребуется ввести пароль пользователя, находящегося в группе **wheel**.

6.2.3 Вкладка «Ресурсы»

В данном окне в реальном времени выводится информация о ресурсах (в виде графиков) – использование процессора (CPU), использование оперативной памяти (RAM) и файла подкачки (SWAP), использование сети;

6.2.4 Вкладка «Файловые системы»

С помощью данной вкладки можно просматривать информацию о всех файловых системах ОС.

Отображается информация о типе файловой системы, общий объём и объём доступного пространства.

6.3 Команды интерпретатора BASH для мониторинга и управления процессами

6.3.1 Команды получения общей информации о системе

\$ echo \$SHELL - информация о текущем интерпретаторе

\$ whoami - информация о текущем пользователе

\$ pwd - информации о текущем каталоге

\$ free - информация об оперативной памяти и файле подкачки

\$ df - информация о afqkjds[cbcntvf[

6.3.2 Команды получения информации о процессах

\$ echo \$\$ - информация об идентификаторе текущего процесса(PID)

\$ echo \$PPID - информация об идентификаторе родительского процесса(PPID)

\$ pidof <имя процесса> - информация об идентификатор процесса по его имени

\$ pgrep -l <имя процесса> - информация об идентификатор процесса по его имени

\$ ps <параметр> - информация о выполняемых процессах

Параметр	Значение
Без параметра	информации о выполняемых процессах текущего пользователя в текущем интерпретаторе команд
-a	отобразить все процессы, связанных с терминалом (отображаются процессы всех пользователей)
-e	отобразить все процессы
-t <список терминалов>	отобразить процессы, связанные с терминалами
-u <имена пользователей>	отобразить процессы, связанные с данными пользователями
-g <имена групп>	отобразить процессы, связанные с данными группами пользователей

-x	отобразить все процессы, не связанные с терминалом
-----------	--

6.3.3 Команды управления процессами

\$ nice - определение текущего значения **nice** по умолчанию

nice [- value] command - изменение значения **nice** при запуске программы.

где **value** - значение (от -20 до +19), добавляемое к значению **nice** процесса-родителя. Полученная сумма будет значением **nice** для запускаемого процесса.

Если параметр **value** не задан, то по умолчанию для процесса-потомка устанавливается значение **nice**, увеличенное на 10 по сравнению со значением **nice** родительского процесса.

renice -n <значение nice> -p <PID> - изменение значения **nice** выполняемого процесса

kill <-номер сигнала> <PID> – принудительное завершение процесса