УДК 004.451.26

Бондарева Станислава Андреевна

студент

РГУ нефти и газа (НИУ) имени И.М. Губкина

Bondareva S.A.

student

Gubkin Russian State University (NRU) of Oil and Gas

Russia, Moscow

Денисюк М.В.

студент

РГУ нефти и газа (НИУ) имени И.М. Губкина

Denisyuk M.V.

student

Gubkin Russian State University (NRU) of Oil and Gas

ПЛАНИРОВЩИКИ ПРОЦЕССОВ НА ОС АЛЬТ

Process schedulers on the Alt OS

***Аннотация**: в работе проанализирована работа планировщиков процессов операционной системы Альт при различном количестве одновременно запущенных потоков программы cpuburn. На основе сравнения таблицы проанализирована нагрузка на процессор и сделаны выводы о каждом алгоритме планирования процессов.*

***Ключевые слова**: планировщик процессов, загруженность системы.*

Abstract: *the paper analyzes the operation of the process schedulers of the Alt operating system with a different number of simultaneously running cpuburn program threads. Based on a comparison of the table, the processor load is analyzed and conclusions are drawn about each process planning algorithm.*

Keywords: *process scheduler, system workload.*

Планировщики процессов являются важной частью работы операционных систем. Они обеспечивают эффективное использование вычислительных ресурсов и повышают производительность системы, распределяя процессорное время между несколькими задачами. С помощью них современные операционные системы способны обрабатывать большое количество параллельных задач. Без них работа на компьютере не была бы такой эффективной.

В зависимости от задач и требований к ним необходимо использовать различные планировщики процессов. Один универсальный планировщик не сможет эффективно справиться со всеми ситуациями.

Объект исследования – процесс и их планирование в операционных системах.

Предмет исследования – сравнительный анализ эффективности алгоритмов планирования процессов операционной системы ALT Linux.

Цель – сравнить работу планировщиков процессов на отечественной операционной системе ALT Linux.

Литературный обзор

На эту тему было уже написано много статей. В пример возьму статью А.Г. Уймина «Моделирование телекоммуникационной сети средствами сетевых инструментов Linux: инструменты создания цифровых двойников» [10]

Если обобщить статьи, то основную информацию можно изложить следующим образом.

В операционных системах процесс - это программа, которая выполняется, используя ресурсы компьютера: память, центральный процессор. Иначе говоря, это активное состояние программы.

Например, пользователь запускает какое-либо приложение, компьютер получает инструкции от программы и выполняет их. Активно работающая версия программы - процесс.

В большинстве случаев пользователи хотят запускать более одной программы одновременно: различные браузеры, игры, текстовые редакторы. Современные операционные системы могут запускать сотни процессов одновременно. Но они только создают такое представление для пользователя. Процессор переключается между программами, предоставляя каждой определенное время (от десятков до сотен миллисекунд). В каждый момент он работает только с одной программой. Реальная параллельная работа программ может осуществляться в многопроцессорных и многоядерных системах. Так четырехъядерная система позволяет одновременно выполнять 4 процесса.

При запуске программа загружается в основную память компьютера, после чего делится на четыре части:

1. Стек. В нем хранится временная информация: вызовы функций, локальные переменные.
2. Куча. В ней хранится память, которая используется во время работы. Например, используется в случаях, когда программе требуется большее количество памяти для хранения дополнительных данных.
3. Раздел данных. Здесь хранятся глобальные и статические переменные, которые программа использует в процессе выполнения.
4. Раздел с текстом. В этой части хранятся фактические инструкции программы для компьютера.

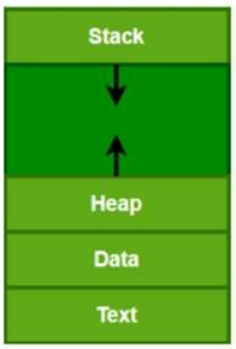

Рисунок 1. Процесс в памяти

У каждого процесса есть атрибуты, которые помогают ОС управлять им и контролировать его. Они включают в себя:

1. Идентификатор процесса (PID), являющийся уникальным номером.
2. Состояние процесса: показывает текущий статус процесса.
3. Информация о планировании: данные, на основе которых ОС решает, какой процесс должен выполняться следующим (уровни приоритета).
4. Информация об устройствах ввода и вывода.
5. Учетная информация: продолжительность выполнения процесса, количество использованного процессорного времени и другие данные об использовании ресурсов.
6. Информация об управлении памятью: сведения о пространстве памяти, которое выделенно процессу.

В системе процесс может находиться в одном из трех основных состояний:

1) Выполнение – активное состояние процесса, во время которого процесс обладает всеми необходимыми ресурсами и выполняется процессором;
2) Ожидание - пассивное состояние процесса, процесс не выполняется по внутренним причинам, он ждет осуществления некоторого события, или освобождения необходимого для него ресурса;

3) Готовность - пассивное состояние процесса, процесс заблокирован из-за внешних причин: имеются все необходимы ресурсы, но пока что выполняется другой процесс.

Рисунок. 2. Граф состояний процесса

Из-за того, что процессор может выполнять только одну программу, то для эффективного использования вычислительных ресурсов необходимо использовать планировщиков процессов. Планирование – это обеспечение поочередного доступа процессов к одному процессору, а планировщик - отвечающая за это часть операционной системы.

Планирование процессов включает в себя определение момента времени для смены выполняемого процесса и выбор процесса на выполнение из очереди готовых процессов;

Различают виды алгоритмов планирования с переключениями и без. Алгоритм планирования без переключений (неприоритетный) не требует прерывание по аппаратному таймеру, процесс останавливается только когда блокируется или завершает работу. Алгоритм с переключениями (приоритетный) требует прерывание по аппаратному таймеру, процесс работает только отведенный период времени, после этого он приостанавливается по таймеру, чтобы передать управление планировщику.

Классы процессов Alt Linux:

1. Процессы реального времени (real_time), обслуживаемые по алгоритму FIFO. Эти процессы имеют наивысшие приоритеты. Они не могут прерываться

другими процессами. Исключением является процесс реального времени, который перешел в состояние готовности.

2. Процессы реального времени (real_time), обслуживаемые в порядке циклической очереди. Отличаются от процессов реального времени FIFO тем, что они могут прерываться по таймеру.

3. Процессы разделения времени (sharing_time). Обслуживаются в режиме пакетной обработки с выделением им определенного времени.

Алгоритмы планирования процессов Alt Linux:

Для того, чтобы узнать, какие планировщики процессов существуют в операционной системе, а также допустимые для них приоритеты можно использовать команду chrt -m

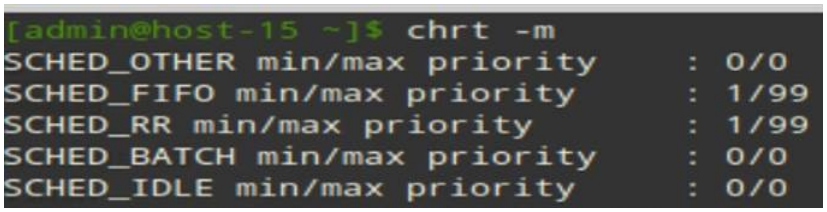

Рисунок. 3. Планировщики процессов в ОС Альт

Таблица 1. Описание планировщиков процессов

Название планировщика	Описание
SCHED_OTHER	Алгоритм планирования на основе разделения времени. Для выполнения выбирается процесс из списка со статическим приоритетом 0. Динамический приоритет основан на значении уступчивости и увеличивается с каждым квантом времени, при котором процесс был готов к работе, но ему было отказано. Таким образом время

	равномерно распределяется между всеми процессами с алгоритмом.
SCHED_FIFO	Принцип планирования - «первым пришел – первым обслужен» без использования квантов времени. Предназначен для класса процессов реального времени, обслуживаемых по алгоритму FIFO.
SCHED_RR	Алгоритм планирования с квантованием. Предназначен для класса процессов реального времени, обслуживаемых в порядке циклической очереди.
SCHED_BATCH	Алгоритм планирования на основе разделения времени с учётом значения приоритета nice. Приоритет процессов ниже, чем у процессов, планируемых на основе SHED_OTHER.
ED_IDLE	Процессам с этой политикой пРисуноквaивается самый низкий приоритет.

С помощью команды top можно просмотреть список текущих процессов. Интерфейс этой команды после запуска выглядит так:

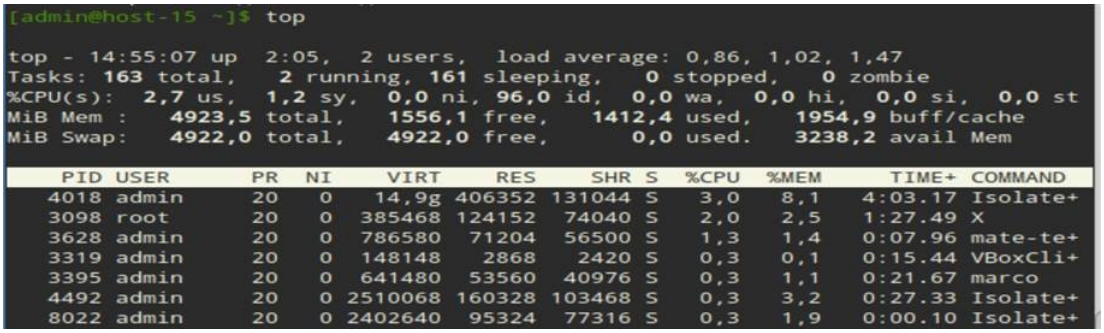

Рисунок. 4. Интерфейс команды top

Верхняя рабочая зона содержит сведения о времени работы сервера, свободных и занятых ресурсах, пользователях, а основная - это динамически обновляемая таблица, которая содержит сведения о процессах.

В верхнем левом углу отображено текущее время, время безотказной работы системы, количество активных сеансов пользователя.

В разделе **Tasks** отображается статистика процессов, выполняемых в системе: общее количество процессов, активные, спящие, остановленные и процессы-зомби.

Раздел использования **CPU** показывает показателя загруженности системы.

- **us (user)** — использование процессора пользовательскими процессами.
- **sy (system)** — использование процессора системными процессами.
- **ni (nice)** — использование процессора процессами с изменённым приоритетом с помощью команды nice.
- **id (idle)** — свободные ресурсы.

Последние 2 строки показывают информацию об использовании памяти в системе. Строки **Mem** и **Swap** отображают информацию о оперативной памяти и области подкачки. Указаны значения общего, свободного, используемого объема и кеша. Avail Mem - это объем памяти, который может быть выделен для процессов, не используя большую область диска.

В таблице используются следующие обозначения:

PID	Это идентификатор процесса, уникальное положительное целое число, которое идентифицирует процесс.
USER	Это эффективное имя пользователя (соответствующее идентификатору пользователя) пользователя, который запустил этот процесс. Linux назначает реальный идентификатор пользователя и эффективный идентификатор пользователя для процессов; последний позволяет процессу действовать от имени другого пользователя. Например, пользователь, не являющийся пользователем root, может с правами root установить пакет.
PR	Поле показывает приоритет выполнения процесса с точки зрения ядра.
NI	Поле показывает nice-значение процесса.
VIRT	Общий объем памяти, потребляемый процессом.Он включает в себя код программы, данные, хранящиеся в памяти, а также любые области памяти, которые были подкачены на диск.
RES	Количество памяти, потребляемая процессом в оперативной памяти.
SHR	Объем памяти, совместно используемый другими процессами.
S	В этом поле отображается состояние процесса в однобуквенной форме(R - Runnable, D - Interruptible sleep, S - Uninterruptible sleep, T - Stopped, Z - Zombie).
%CPU	Параметр выражает объем в процентах от общей доступной оперативной памяти ОЗУ.
%MEM	Параметр выражает значение RES в процентах от общей доступной оперативной памяти.
TIME+	Общее время процессора, используемое процессом с момента его начала, с точностью до сотых долей секунды.
COMMAND	Здесь отображено название процессов.

Рисунок 5. Обозначения в таблице

Экспериментальная часть

Методика исследования: на ОС ALT Linux буду запускать программу cpuburn, последовательно изменяя используемый планировщик процесса и анализируя загруженность системы. Так как рассматриваемая программа является однопоточной, то исследование нужно проводить при запуске 1/4/7 потоков одновременно.

1) Анализ при одном включенном потоке

Включим cpuburn с помощью команды burnP6. Воспользуемся командой top, чтобы просмотреть список текущих процессов.

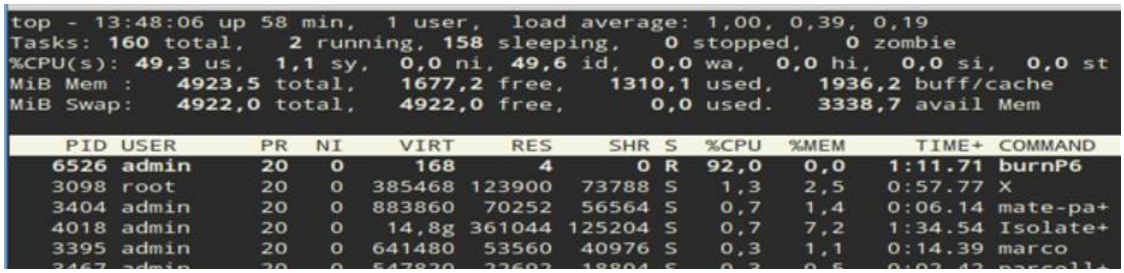

Рисунок. 6. Список процессов при запуске одного потока программы

Теперь, чтобы получить текущую политику планирования и приоритет для процесса, вызванным командой burnP6, используем chrt следующим образом

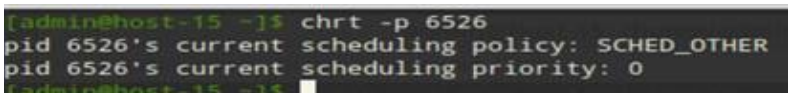

Рисунок 7. Просмотр текущей политики

Замечаем, что по умолчанию используется алгоритм планирования **SCHED_OTHER** с приоритетом 0.

Теперь, чтобы изменить политику на **SCHED_FIFO**, нужно обладать правами суперпользователя (используем команду su-). Далее пишем команду chrt -f -p <PID> и проверяем, какая политика планирования станет использоваться для процесса.

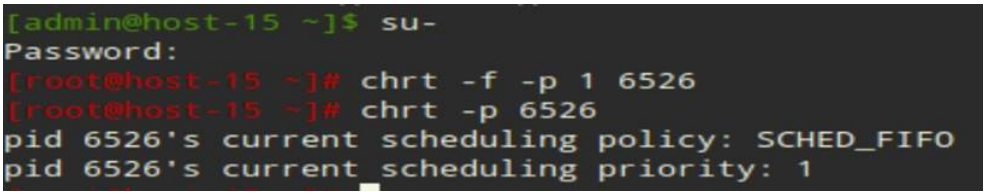

Рисунок 8. Смена политики

После этого посмотрим список текущих процессов.

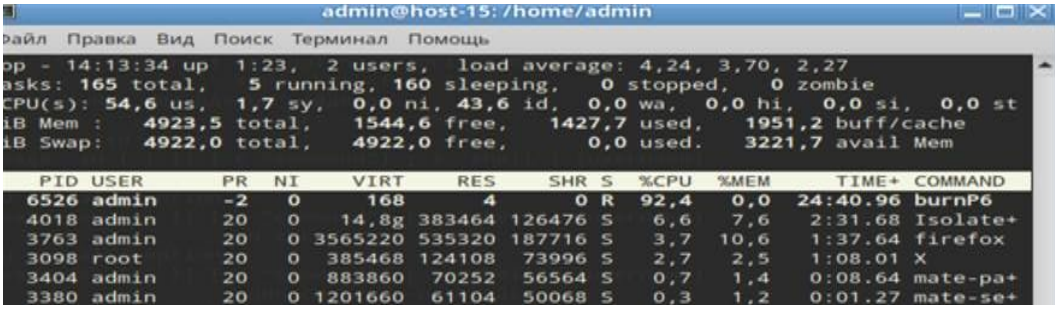

Рисунок 9. Список процессов

Для применения алгоритма **SCHED_RR** используем команду chrt -r -p <PID>

```
[root@host-15 ~]# chrt -r -p 1 6526
[root@host-15 ~]# chrt -p 6526
pid 6526's current scheduling policy: SCHED_RR
pid 6526's current scheduling priority: 1
```

Рисунок 10. Смена политики

Рисунок 11. Список процессов

Теперь, чтобы изменить политику на **SCHED_BATCH** используем следующую команду: chrt -b -p <PID>

```
[root@host-15 ~]# chrt -b -p 0 6526
[root@host-15 ~]# chrt -p 6526
pid 6526's current scheduling policy: SCHED_BATCH
pid 6526's current scheduling priority: 0
```

Рисунок 12. Смена политики

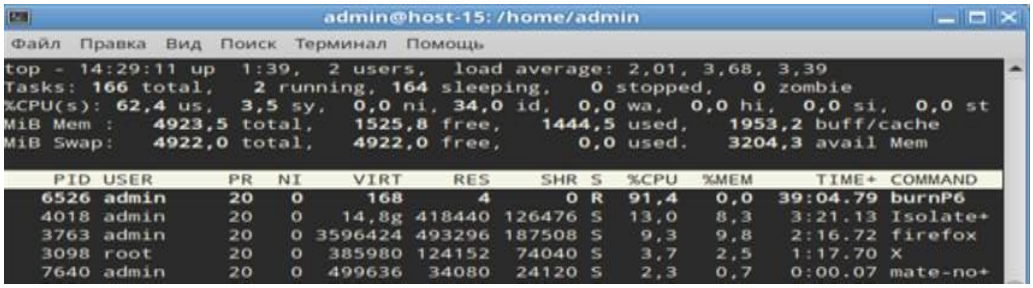

Рисунок 13. Список процессов

Для **SCHED_IDLE** пишем chrt -i -p <PID>

```
[root@host-15 ~]# chrt -i -p 0 6526
[root@host-15 ~]# chrt -p 6526
pid 6526's current scheduling policy: SCHED_IDLE
pid 6526's current scheduling priority: 0
```

Рисунок 14. Смена политики

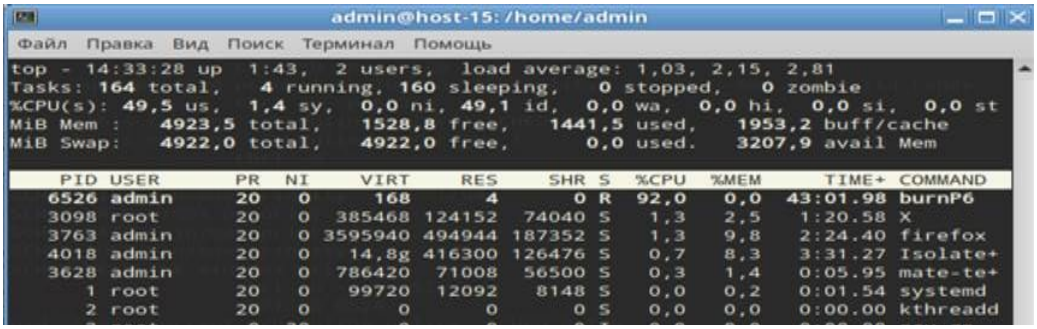

Рисунок 15. Список процессов

2) Анализ с четырьмя запущенными потоками

По аналогии с прошлым пунктом запустим четыре потока программы cpuburn и воспользуемся командой top.

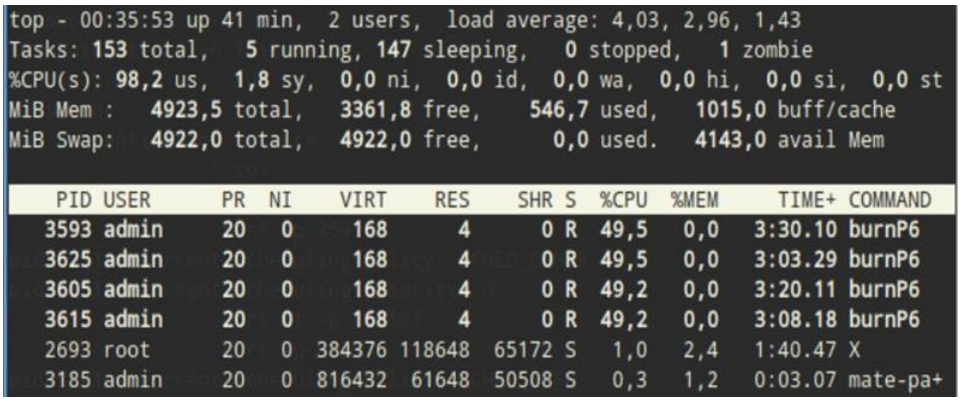

Рисунок 16. Список процессов

Проверим какой планировщик установлен по умолчанию. Как и в прошлом эксперименте используется **SCHED_OTHER** с приоритетом 0.

```
[root@host-15 ~]# chrt -p 3605
pid 3605's current scheduling policy: SCHED_OTHER
pid 3605's current scheduling priority: 0
```

Рисунок 17. Текущая политика

Изменяем политику на **SCHED_FIFO**, используя команду chrt -f -p <PID> и смотрим список текущих процессов.

```
[root@host-15 ~]# chrt -f -p 1 3605
[root@host-15 ~]# chrt -p 3605
pid 3605's current scheduling policy: SCHED_FIFO
pid 3605's current scheduling priority: 1
```

Рисунок 18. Смена политики

```
top - 00:36:26 up 41 min,  2 users,  load average: 4,16, 3,11, 1,54
Tasks: 153 total,   6 running, 146 sleeping,   0 stopped,   1 zombie
%CPU(s): 97,0 us,  1,8 sy,  0,0 ni,  1,2 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   4923,5 total,   3372,8 free,    535,7 used,   1015,0 buff/cache
MiB Swap:   4922,0 total,   4922,0 free,      0,0 used.   4154,0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3605 admin     -2   0     168      4      0 R  95,0   0,0   3:41.99 burnP6
   3593 admin     20   0     168      4      0 R  34,6   0,0   3:44.54 burnP6
   3615 admin     20   0     168      4      0 R  34,2   0,0   3:22.55 burnP6
   3625 admin     20   0     168      4      0 R  30,9   0,0   3:17.33 burnP6
   2693 root      20   0  384376 118648  65172 S   1,3   2,4   1:41.06 X
```

Рисунок 19. Список процессов

Применяем политику **SCHED_RR** с помощью команды chrt -r -p <PID>.

```
[root@host-15 ~]# chrt -r -p 1 3605
[root@host-15 ~]# chrt -p 3605
pid 3605's current scheduling policy: SCHED_RR
pid 3605's current scheduling priority: 1
```

Рисунок 20. Смена политики

```
top - 00:37:50 up 43 min,  2 users,  load average: 4,50, 3,48, 1,81
Tasks: 153 total,   6 running, 146 sleeping,   0 stopped,   1 zombie
%CPU(s): 96,3 us,  2,5 sy,  0,0 ni,  1,2 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   4923,5 total,   3372,8 free,    535,6 used,   1015,0 buff/cache
MiB Swap:   4922,0 total,   4922,0 free,      0,0 used.   4154,1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3605 admin     -2   0     168      4      0 R  95,3   0,0   5:01.93 burnP6
   3593 admin     20   0     168      4      0 R  32,6   0,0   4:13.61 burnP6
   3615 admin     20   0     168      4      0 R  31,9   0,0   3:51.55 burnP6
   3625 admin     20   0     168      4      0 R  27,9   0,0   3:44.11 burnP6
   2693 root      20   0  384376 118648  65172 S   5,3   2,4   1:41.84 X
```

Рисунок 21. Список процессов

Для применения алгоритма **SCHED_BATCH** используем команду chrt -b -p <PID>

```
[root@host-15 ~]# chrt -b -p 0 3605
[root@host-15 ~]# chrt -p 3605
pid 3605's current scheduling policy: SCHED_BATCH
pid 3605's current scheduling priority: 0
```

Рисунок 22. Смена политики

```
top - 00:39:21 up 44 min,  2 users,  load average: 4,34, 3,71, 2,05
Tasks: 153 total,   5 running, 147 sleeping,   0 stopped,   1 zombie
%CPU(s): 98,3 us,  1,7 sy,  0,0 ni,  0,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   4923,5 total,   3381,1 free,    527,3 used,   1015,0 buff/cache
MiB Swap:   4922,0 total,   4922,0 free,      0,0 used.   4162,4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3593 admin     20   0     168      4      0 R  50,2   0,0   4:52.28 burnP6
   3625 admin     20   0     168      4      0 R  49,5   0,0   4:21.41 burnP6
   3605 admin     20   0     168      4      0 R  48,8   0,0   6:05.25 burnP6
   3615 admin     20   0     168      4      0 R  48,8   0,0   4:29.92 burnP6
   2693 root      20   0  384376 118648  65172 S   1,0   2,4   1:42.70 X
   3185 admin     20   0  816432  61648  50508 S   0,7   1,2   0:03.63 mate-pa+
```

Рисунок 23. Список процессов

Для **SCHED_IDLE** пишем chrt -i -p <PID>

```
[root@host-15 ~]# chrt -i -p 0 3605
[root@host-15 ~]# chrt -p 3605
pid 3605's current scheduling policy: SCHED_IDLE
pid 3605's current scheduling priority: 0
```

Рисунок 24. Смена политики

```
top - 00:40:18 up 45 min,  2 users,  load average: 4,13, 3,77, 2,17
Tasks: 152 total,   5 running, 146 sleeping,   0 stopped,   1 zombie
%CPU(s): 97,7 us,  2,3 sy,  0,0 ni,  0,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   4923,5 total,   3380,7 free,    527,8 used,   1015,0 buff/cache
MiB Swap:   4922,0 total,   4922,0 free,      0,0 used.   4161,9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3593 admin     20   0     168      4      0 R  49,8   0,0   5:20.49 burnP6
   3615 admin     20   0     168      4      0 R  49,8   0,0   4:58.18 burnP6
   3625 admin     20   0     168      4      0 R  48,8   0,0   4:49.53 burnP6
   3605 admin     20   0     168      4      0 R  48,2   0,0   6:33.46 burnP6
   2693 root      20   0  384376 118648  65172 S   1,3   2,4   1:43.37 X
```

Рисунок 25. Список процессов

3) Анализ с семью запущенными потоками

Проверяем список текущих процессов и узнаем политику, установленную по умолчанию - **SCHED_OTHER** с приоритетом 0.

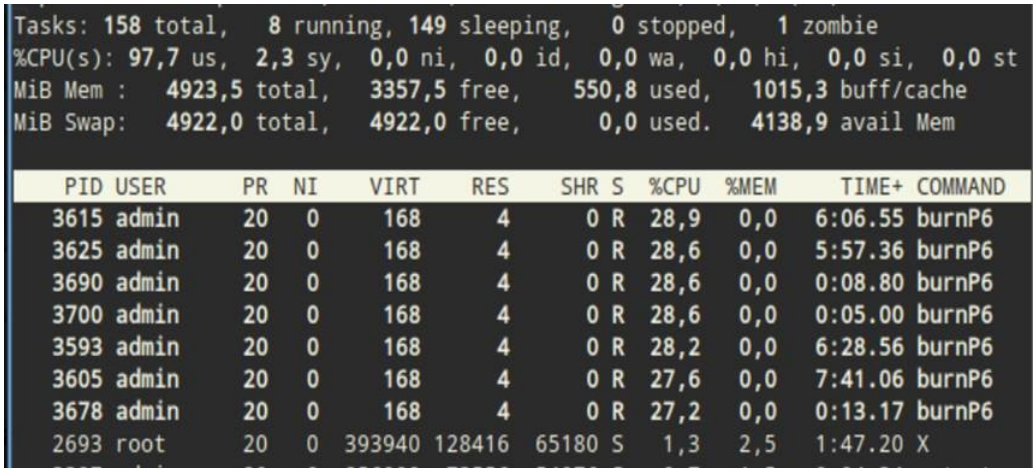

Рисунок 26. Список процессов

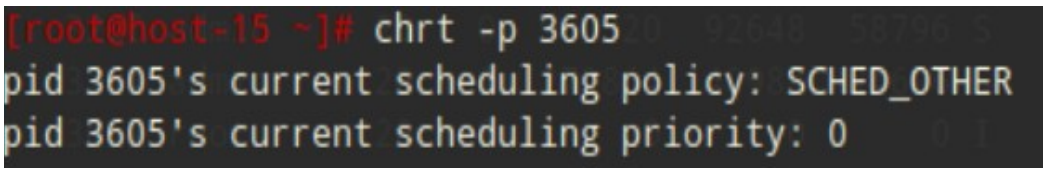

Рисунок 27. Текущая политика

Изменяем используемую политику на **SCHED_FIFO** с использованием команды из прошлого пункта при работе с этой политикой. Проверяем список запущенных процессов.

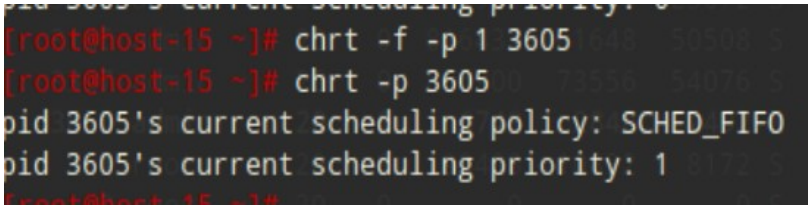

Рисунок 28. Смена политики

Рисунок 29. Список процессов

Применяем политику алгоритма **SCHED_RR,** используем команду chrt -r -p <PID>

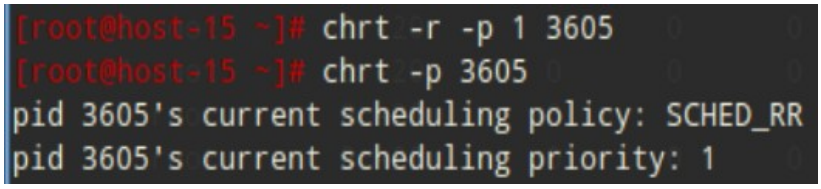

Рисунок 30. Смена политики

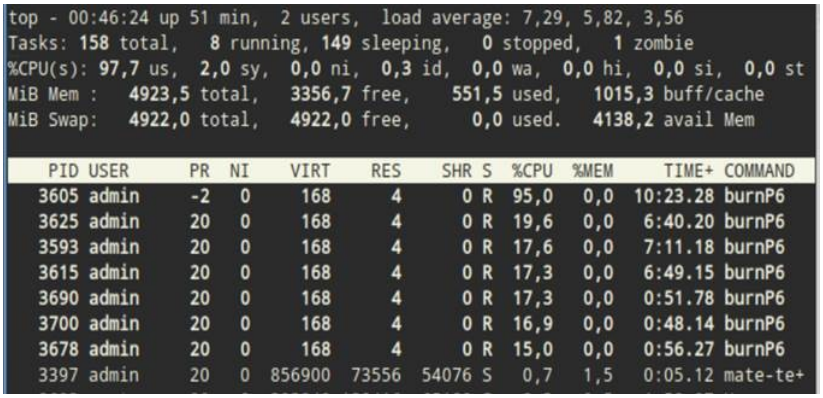

Рисунок 31. Список процессов

Работа с политикой на **SCHED_BATCH** осуществляется с использованием следующей команды: chrt -b -p <PID>

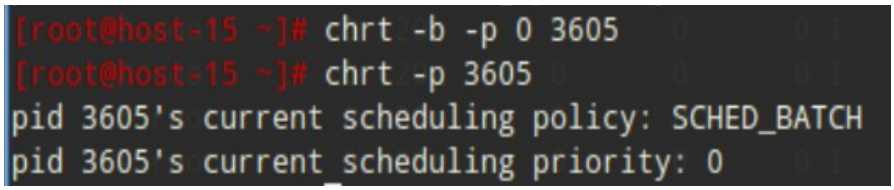

Рисунок 32. Смена политики

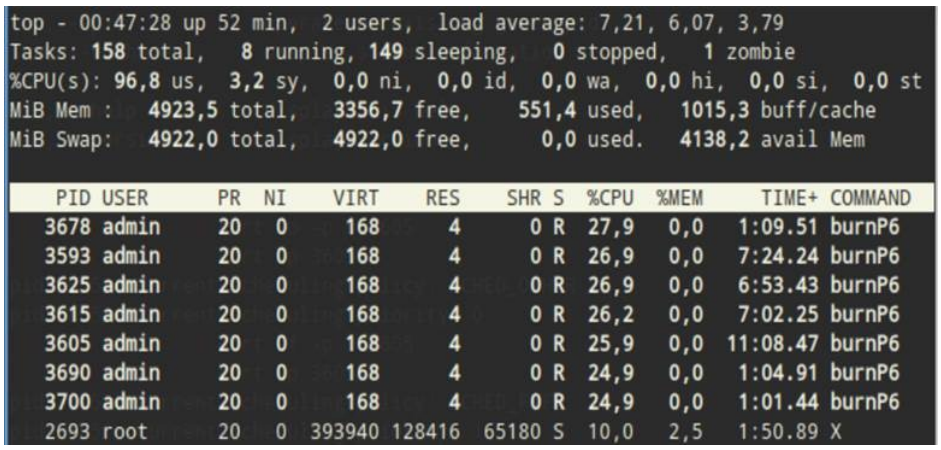

Рисунок 32. Список процессов

Для **SCHED_IDLE** пишем chrt -i -p <PID>

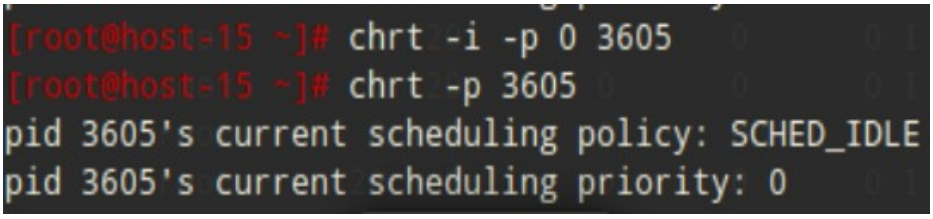

Рисунок 34. Смена политики

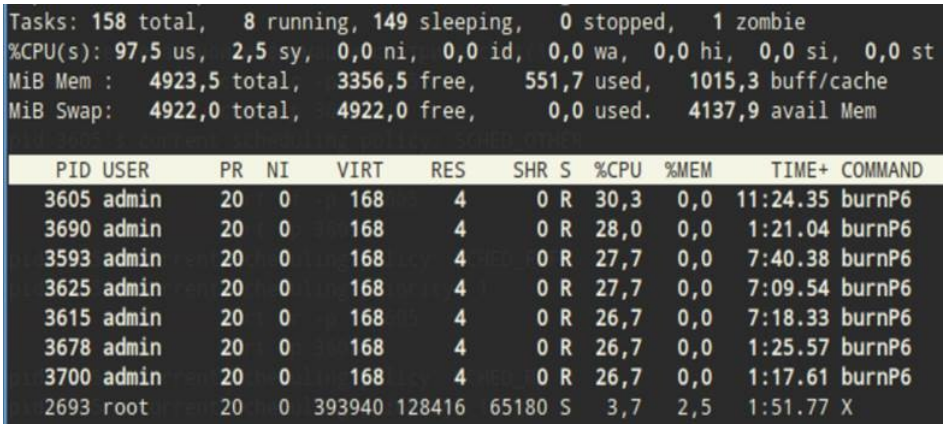

Рисунок 35. Список процессов

Составим таблицу с различающимися данными.

Таблица 2. Сравнение загруженности системы при разном количестве потоков

Название политики	us			sy			id			Используемая память		
Кол-во потоков	1	4	7	1	4	7	1	4	7	1	4	7
SCHED_OTHER	49,3	98,2	98	1,1	1,	2	49,6	0	0	1310,1	546,	550,8

					8						7	
SCHED_FIFO	54,6	97	98	1,7	1,8	2	43,6	1,2	0,3	1427	535,7	551,2
SCHED_RR	53,1	96,3	98	1,2	2,5	2	45,7	1,2	0,3	1463	535,6	551,5
SCHED_BATCH	62,4	98,3	97	3,5	1,7	3	34	0	0	1444,5	527,3	551,4
SCHED_IDLE	49,5	97,7	98	1,4	2,3	3	49,1	0	0	1441,5	527,8	1015

Вывод

Политики **SCHED_OTHER** показывает относительно сбалансированное использование ресурсов: использование процессора самое низкое, остается наибольшее количество свободных ресурсов.

При использовании политики **SCHED_FIFO** использование процессора пользовательскими процессами значительно выше, чем у SCHED_OTHER при однопоточном исследовании. Это может указывать на то, что процессы реального времени (FIFO) захватывают значительную часть процессорного времени, оставляя мало ресурсов для других процессов. При анализе с 4/7 потоками система работает незначительно, но лучше, чем при использовании SCHED_OTHER.

SCHED_RR - циклическое квантование работает относительно эффективно, обеспечивая более равномерное распределение процессорного времени, чем SCHED_FIFO. В сравнении с другими алгоритмами загруженность процессора пользовательскими процессами меньше.

SCHED_BATCH – низкая загрузка id и высокая загрузка пользователя (us). Процессы с низким приоритетом, работают эффективно с точки зрения пользовательского времени, но получают мало времени процессора, что проявляется в низкой системной загрузке (sy).

SCHED_IDLE - низкая загрузка процессора только при однопоточном анализе.

Выбор политики планирования зависит от конкретных задач. Например, SCHED_OTHER и SCHED_RR показывают лучшую масштабируемость при увеличении количества потоков. SCHED_BATCH неэффективен при маленькой нагрузке, SCHED_IDLE – при большой.

В целом, SCHED_OTHER подходит для большинства интерактивных приложений, SCHED_FIFO и SCHED_RR — для задач реального времени, SCHED_BATCH — для фоновых задач с низким приоритетом, а SCHED_IDLE предназначен для процессов, которые выполняются только при отсутствии другой работы.

Литература:

1. ALT Linux - sisyphus - cpuburn-1.4a-alt1 - CPU testing utilities - [Электронный ресурс] - URL: https://packages.altlinux.org/en/sisyphus/srpms/cpuburn/ (дата обращения: 10.12.24).
2. chrt command in Linux with examples – GeeksforGeeks - [Электронный ресурс] - URL: https://translated.turbopages.org/proxy_u/en-ru.ru.b9e09dfb-6758b48b-6d3b21a7-74722d776562/https/www.geeksforgeeks.org/chrt-command-in-linux-with-examples/ (дата обращения: 10.12.24).
3. Operating Systems: Three Easy Pieces. Part 2: Абстракция: Процесс (перевод) / Хабр - [Электронный ресурс] - URL: https://habr.com/ru/articles/446866/ (дата обращения: 11.12.24).
4. States of a Process in Operating Systems – GeeksforGeeks – [Электронный ресурс] - URL: https://www.geeksforgeeks.org/states-of-a-process-in-operating-systems/ (дата обращения: 11.12.24).

5. Как управлять процессами в Linux - [Электронный ресурс] - URL: https://1cloud.ru/help/security/prosmotr-i-upravlenie-protsessami-linux-s-pomoshhyu-top (дата обращения: 11.12.24).
6. Кручинин А.Ю. К 84 Операционные системы [Электронный ресурс] : учебное пособие / А.Ю. Кручинин, Р.Р. Галимов., А.А. Рычкова – Оренбург: ОГУ, 2019. – 152 с. (дата обращения: 09.12.24).
7. ОС: 5.1 Лекция- Планирование процессов. | Сайт дистанционного образования - MOODLE КНИТУ (КХТИ)- [Электронный ресурс] - URL: https://moodle.kstu.ru/mod/page/view.php?id=55 (дата обращения: 09.12.24).
8. Сёмкин П.С., Сёмкин А.П. Методические материалы к лабораторным работам по дисциплине «Операционные системы» «ОС Alt Linux. Мониторинг и управление процессами» (дата обращения: 11.12.24).
9. Управление процессами - [Электронный ресурс] - URL: https://dit.isuct.ru/IVT/BOOKS/OPERATING_SYSTEMS/OPER12/GLAVA_6.HTM (дата обращения: 11.12.24).
10. Уймин, А. Г. Моделирование телекоммуникационной сети средствами сетевых инструментов Linux: инструменты создания цифровых двойников / А. Г. Уймин, О. Р. Никитин // I-methods. – 2023. – Т. 15, № 2. – EDN NFJDVH. (дата обращения: 09.12.24).