Шпора по осям

**Stress-ng**

Метрики:

*Bogo ops* – количество бого-операций, которое выполняет каждый стрессор

*Real time* – реальное время, затраченное на выполнение стрессора

*Usr time* – время, затраченное на выполнение в пользовательском режиме, на прямую к ядру не обращается, суммируется со всех потоков (если потоков больше одного – *usr time* будет больше *real time*)

*Sys time* – время, которое код выполняется в ядре

*Bogo ops/s* – количество бого-операций в секунду

*CPU used per instance* – процент использования процессорного времени стрессором

*RSS Max* – Resident Set Size – максимальный объем занимаемой стрессором физической памяти

## Блок 1. CPU

--*cpu N* – запускает N процессов тестирования процессора разными методами. Вместо последовательного выполнения всех методов нагрузки можно выполнить какой-то определенный с помощью –cpu-method

**Top**

-d (интервал обновления) -H (показ информации по каждому потоку)

Метрики:

*Top:* количество времени, количество пользователей с активными потоками, *load average* (среднее количество процессов в очереди на выполнение в системе за 5, 10 и 15 минут)

*Threads: total* (общее количество потоков), *running* (количество выполняющихся потоков), *sleeping* (спящие процессы), *stopped* (остановленные процессы), *zombie* (потоки, которые завершили работу, но не были полностью завершены. Они остаются в системе, ожидая, чтобы их родительские процессы "забрали" их завершенный статус. Это может произойти, когда родительский процесс не выполнил определенное действие для завершения потомка. В общем, зомби-процессы не вызывают проблем, но их накопление может потребовать вмешательства для удаления их из системы)

*%Cpu: us* (user - процент времени выполнения пользовательских процессов), *sy* (system - процент времени выполнения системных процессов), *ni* (nice - процент времени выполнения задач с измененным приоритетом), *id* (idle - процент времени простоя процессора), *wa* (wait - процент времени ожидания процессов ввода-вывода), *hi* (hardware interrupts - процент времени обработки аппаратных прерываний), *si* (software interrupts - процент времени обработки программных прерываний), *st* (steal - процент времени, украденного виртуализацией (создание изолированных окружений в рамках одного физ.устройства – экономия ресурсов, безопасность и тд))

*MiB Swap* обозначает количество оперативной памяти, использованной в качестве пространства подкачки (swap) в мегабайтах (MiB). Подкачка позволяет системе выгружать часть неиспользуемых данных из оперативной памяти на диск, освобождая место для активных задач.

Когда оперативной памяти не хватает для выполнения всех активных процессов, операционная система может использовать область на жестком диске (swap) в качестве временного хранилища для частей данных, которые редко используются или вообще не используются в данный момент.

*MiB Mem* указывает на количество физической оперативной памяти (RAM), используемой системой в мегабайтах (MiB). Эта метрика показывает общий объем оперативной памяти, используемой системой для выполнения процессов и задач в данный момент времени.

*PID* – идентификатор процесса (Process ID)

*User* – имя пользователя

*PR* – приоритет процесса

*NI* – значение Nice, которое может менять приоритет процесса

*Virt* - общий объем виртуальной памяти, используемый процессом.

*Res* - резидентная память (Resident Set Size), количество физической памяти, используемое процессом в данный момент.

*SHR* - объем разделяемой памяти, используемой процессом (shared).

*S* - статус процесса (running - R, sleeping - S, stopped - T, zombie - Z, и т. д.).

*%CPU* - процент процессорного времени, используемого процессом.

*%Mem* - процент физической памяти, используемой процессом.

*Time+* - общее время, которое процесс занимает процессорное время начиная с последнего обновления экрана top.

*Command* – название команды

## Блок2. Cache

"Buffers" представляют собой ту часть оперативной памяти, которая выделена под кэширование блоков диска. "Cached" похож на "Buffers", но в этом случае кэшируются страницы, прочитанные из файлов.

Определения:

*Буферы* - это временное хранилище для сырых блоков диска, то есть данные кэшируются для записи на диск, обычно не очень большие (примерно 20 МБ). Таким образом, ядро может централизовать рассеянные записи и оптимизировать их равномерно. Например, несколько небольших записей могут быть объединены в одну большую и т. д.

*Кэш* - это кэш страниц для чтения файлов с диска, который используется для кэширования данных, прочитанных из файлов. Это позволяет в следующий раз быстро извлечь эти файловые данные непосредственно из памяти, без необходимости снова обращаться к медленному диску.

Buffers - память, используемая буферами ядра (Buffers в /proc/meminfo)

Cache - память, используемая кэшем страниц и slabs (В контексте системы Linux, "slabs" (слабики) представляют собой структуры данных, используемые для управления выделением памяти ядра. Они служат для упорядочения и оптимизации выделения памяти, позволяя эффективно управлять памятью ядра для различных структур данных и объектов.)

**Кэш страниц** - это основной дисковый кеш, используемый ядром Linux. В большинстве случаев ядро обращается к кешу страниц при чтении или записи на диск. … Если имеется достаточно свободной памяти, страница хранится в кеше в течение неопределенного периода времени, а затем может быть повторно использована другими процессами без доступа к диску.

В Linux кэш страниц **ускоряет множество обращений к файлам в энергонезависимой памяти**. Это происходит потому, что при первом чтении или записи на носители данных, такие как жесткие диски, Linux также сохраняет данные в неиспользуемых областях памяти, которые действуют как кеш.

**Буферы** - это буферы блочного ввода-вывода в памяти. Они относительно недолговечны. До версии ядра Linux 2.4 в Linux были отдельные кеши страниц и буферов. Начиная с версии 2.4, страничный и буферный кеши унифицированы, а Buffers - это **необработанные блоки диска, не представленные в страничном кэше**, т. Е. Не данные файла.

В кэше хранятся данные и инструкции к ним.

*Кэш 1 уровня* расположен близко к логическим блокам, работает на одной скорости с ними. Выделено 32 Кб под данные и 32 Кб под инструкции. В кэше 1 уровня для инструкций хранятся инструкции, готовые к разделению на более мелкие микрооперации (L0 – 1500 микроопераций).

*Кэш 2 уровня* больше по объему (около 512 Кб), чем кэш 1 уровня, требует примерно в 2 раза больше времени для выполнения операций. К нему могут получать доступ несколько ядер (но не все), такие ядра составляют кластеры. Каждое ядро имеет личный набор кэшей 1 и 2 уровня.

*Кэш 3 уровня* расположен рядом с одним ядром, но является общим для всех ядер. Его объем от 2 до 32 Мб.

Наличие нескольких уровней кэша помогает увеличить производительность.

**Когерентность кэша** – свойство кэша, разделенного на уровни. Оно подразумевает целостность данных, которые хранятся в данных разделяемого ресурса. Это свойство помогает кэшам работать согласованно и избегать противоречий. Примеры протоколов обеспечения когерентности: *MESI (Modified, Exclusive, Shared, Invalid), MSI (Modified, Shared, Invalid), MOESI (Modified, Owner, Exclusive, Shared, Invalid).*

**Атрибут –perf** у команды stress-ng ДЛЯ КЭША

Это еще не перф:

- **cache ops per second**: Общее количество операций с кэшем в секунду.
- **shared cache reads per second**: Количество операций чтения из общего кэша в секунду.
- **shared cache writes per second**: Количество операций записи в общий кэш в секунду.

**А это уже перф**

1. **CPU Clock / Task Clock**: Измеряет количество процессорных циклов, использованных задачей. Отображается в байтах в секунду.
2. **Page Faults Total / Minor / Major**: Общее количество ошибок страниц памяти (faults) / Минорные (ошибки, обработанные без обращения к диску) / Мажорные (требующие обращения к диску).
3. **Context Switches / Cgroup Switches**: Количество переключений контекста между процессами / Количество переключений cgroup (группами контроля ресурсов) контекста. *** Cgroups (Control Groups) - это механизм в ядре Linux, который позволяет ограничивать и управлять ресурсами,

такими как CPU, память, I/O и другие, для группы процессов. Он используется для создания изолированных окружений (контейнеров) в операционной системе Linux.***

4. **CPU Migrations**: Количество миграций процессора между различными ядрами.

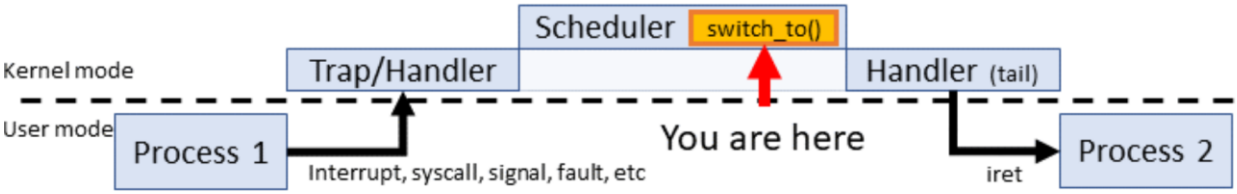
   *Alignment Faults* - это прерывания, возникающие при обращении к памяти или выполнении инструкций, которые требуют выравнивания данных по определенным границам. Если данные не выровнены правильно, CPU генерирует это исключение. Обработка выравнивания данных требует дополнительного времени и ресурсов процессора.

   *Emulation Faults* указывает на ошибки, которые могут возникнуть в процессе эмуляции аппаратных функций или инструкций. Эти ошибки связаны с эмуляцией аппаратных действий на программном уровне, что может привести к прерываниям в работе программы или потребовать дополнительных ресурсов системы для обработки.

5. **System Call Enter / Exit**: Количество входов / выходов из системных вызовов в секунду.
6. **TLB Flushes**: Количество операций сброса Translation Lookaside Buffer. *** Translation Lookaside Buffer (TLB) - это специальный кэш-буфер, используемый процессором для ускорения процесса преобразования виртуальных адресов в физические адреса.***
7. **Kmalloc / Kfree**: Количество выделений / освобождений памяти ядру Linux.
8. **MM Page Alloc / Free**: Количество выделений / освобождений страниц памяти ядра.
9. **MMAP lock start / release / acquire**: Количество блокировок / освобождений / захватов MMAP. *** MMAP обозначает отображение файлов в память (Memory-Mapped Files) в контексте UNIX-подобных операционных систем. Этот механизм позволяет приложениям работать с файлами напрямую через оперативную память, обеспечивая более быстрый доступ к данным.***
10. **RCU Utilization** указывает на использование Read-Copy Update (RCU) в ядре операционной системы. RCU - это механизм синхронизации данных в ядре Linux, позволяющий обеспечить безопасное чтение данных, даже когда они изменяются другими процессами. Эта метрика указывает на количество раз, когда RCU используется для доступа к данным.

11. **RCU Stall Warning** (предупреждения о блокировках RCU) - это показатель количества предупреждений, возникших в связи с событиями, когда процессу пришлось ждать завершения операций RCU.
12. **Sched Migrate Task** - это количество миграций задач, когда задача переносится с одного процессора на другой. Миграция задачи происходит по различным причинам, например, для улучшения балансировки нагрузки между процессорами.
13. **Sched Move NUMA** - это метрика, отражающая количество операций перемещения задачи между узлами NUMA (Non-Uniform Memory Access) в системе, где доступ к памяти различается в зависимости от того, находится ли память на том же узле, что и процессор.
14. **Sched Wakeup / Switch**: Количество разблокировок / переключений планировщика задач.
15. **Soft IRQ Entry / Exit**: Количество входов / выходов в Soft IRQ обработчик. *** Soft IRQ (Software Interrupt Request) - это механизм обработки прерываний в ядре операционной системы Linux. Soft IRQ представляют собой способ, с помощью которого ядро Linux может отложить обработку некоторых событий, чтобы не прерывать выполнение прерываемой обработки (interrupt context) - контекста прерывания.***
16. **NMI handler** отражает количество обработанных обработчиков Non-Maskable Interrupts (NMI), специальных прерываний, которые обрабатываются независимо от состояния флага разрешения прерываний (interrupt-enable flag).
17. **Block BIO Complete** указывает на количество завершенных блочных операций ввода-вывода.
18. **IOMAP Read Page** и **IOMAP Write Page** относятся к операциям чтения и записи страницы ввода-вывода IOMAP.
19. **IO uring submit** и **IO uring complete** связаны с операциями передачи и завершения запросов ввода-вывода через механизм IO_uring.
20. **Writeback Dirty Inode** указывает на количество обратных записей (запись на диск измененных данных) для "грязных" инодов (inode), которые были изменены в памяти.
21. **Writeback Dirty Folio** - аналогично предыдущей метрике, но для измененных страниц (folio) в памяти.
22. **Migrate MM Pages** - операции миграции страниц памяти.
23. **SKB Consume** и **SKB Kfree** - операции относящиеся к работе с сетевыми пакетами (SKB - Socket Buffer).

24. **Lock Contention Begin / End**: Количество начатых / завершенных контендирований блокировки. *** Контендирование блокировки (Lock Contention) происходит, когда несколько процессов или потоков пытаются одновременно получить доступ к общему ресурсу, который находится под защитой блокировки (lock), но могут быть вынуждены ждать друг друга из-за того, что доступ к ресурсу заблокирован другим процессом или потоком. Это приводит к ситуации, известной как блокировка конкуренции (lock contention).***
25. **Maple Tree Read / Write**: Количество чтений / записей в структуру данных Maple Tree. *** Maple Tree (Кленовое дерево) - это структура данных, используемая в ядре операционной системы Linux для представления процессов в системе. Эта структура хранит информацию о процессах и позволяет быстро и эффективно осуществлять доступ к ним и управлять ими. Maple Tree обычно используется внутри планировщика задач ядра Linux для управления процессами и потоками. Это дерево состоит из узлов, представляющих процессы, и содержит информацию о каждом процессе, такую как идентификатор процесса (PID), состояние процесса, приоритет и другие характеристики.***
26. **MSR read / write**: Количество чтений / записей в регистр модели специфичного процессора.
27. **IOMMU IO Page Fault** указывает на количество ошибок (page faults), которые произошли в результате операций ввода-вывода, управляемых IOMMU (Input-Output Memory Management Unit).
28. **IOMMU Map** и **IOMMU Unmap** - операции отображения (mapping) и удаления (unmapping) виртуального адресного пространства на физическое памяти, выполненные через IOMMU.
29. **Filemap page-cache add / del**: Количество добавлений / удалений страниц в файловом кэше.
30. **OOM Compact Retry** и **OOM Wake Reaper** относятся к различным операциям, выполняемым в случае возникновения событий Out-Of-Memory (выход за границы доступной памяти) в системе.
31. **OOM Score Adjust Update**: Количество обновлений оценки OOM (Out Of Memory) Score. *** Out-Of-Memory (OOM) используется в ядре Linux для определения порядка, в котором различные процессы или задачи будут завершены, когда возникает ситуация нехватки памяти.***
32. **Thermal Zone Trip** связан с событиями, связанными с температурными зонами в системе, например, срабатывание термостата.




Переключение контекста происходит с помощью макроса switch_to(), которым управляет schedular (планировщик)

Краткий список задач переключателя контекста:

1. Переуказание рабочего пространства: восстановление стека (SS:SP).
2. Поиск следующей инструкции: восстановление IP (CS:IP).
3. Восстановление состояния задачи: восстановление регистров общего назначения.
4. Своппинг адресных пространств памяти: обновление каталога страниц (CR3)
5. … и многое другое: FPU, структуры данных ОС, регистры отладки, аппаратные обходные пути и т. д.

Переключение контекста в ядре Linux осуществляется с помощью процедуры сохранения текущего контекста процесса и загрузки нового контекста другого процесса. Для этого ядро использует механизм переключения задач (task switch), который включает в себя следующие шаги:

1. Сохранение состояния регистров процессора текущего процесса в его PCB (Process Control Block).
2. Сохранение указателя на вершину стека текущего процесса.
3. Загрузка регистров процессора и указателя на стек нового процесса из его PCB.
4. Переключение контекста на новый процесс.

Переключение контекста может происходить из-за различных событий, таких как смена процессора, завершение текущего процесса, прерывание по таймеру и т.д. Эти события могут быть инициированы как внешними, так и внутренними процессами в системе.

## Блок 3. IO

### Метрики iotop

1. **TID (Thread ID)**: Идентификатор потока или задачи.
2. **PRIO (Priority)**: Приоритет процесса/потока.
3. **USER**: Имя пользователя, от имени которого выполняется процесс.
4. **DISK READ**: Общее количество данных, прочитанных с диска процессом/потоком.
5. **DISK WRITE**: Общее количество данных, записанных на диск процессом/потоком.
6. **SWAPIN**: Количество данных, записанных из области подкачки (swap space) на диск для этого процесса/потока.
7. **IO>**: Общее количество операций ввода-вывода, выполненных процессом/потоком.
8. **COMMAND**: Команда или название процесса/потока.

## Блок 4. Memory

Память может быть выровненная (адреса данных начинаются с адреса, кратного размеру данных, (например, байту, слову, или более крупным границам)) и невыровненная (адрес не кратен размеру данных)

- Выровненная память:
    - Адрес 0x0000: Данные выровнены по границе 4 байта.
    - Адрес 0x0004: Также данные выровнены.
    - Адрес 0x0008: Опять же, данные выровнены по 4 байта.
- Невыровненная память:
    - Адрес 0x0001: Невыровненный адрес для 4-байтовых данных.
    - Адрес 0x0003: Также невыровненный для данных, размером в 4 байта.

**Физическая память** (или ***«ОЗУ»***, ***«RAM»***, ***«оперативка»***) — это энергозависимая память, установленная в компьютере. Для её работы требуется непрерывный поток электричества. Перебои с электропитанием или внезапное выключение компьютера могут привести к стиранию хранящихся в ней данных. Кроме того, эта память является линейно адресуемой. Другими словами, значения адресов памяти увеличиваются линейным образом.

Запуская и исполняя программы, процессор напрямую обращается к физической памяти. Обычно программы хранятся на жестком диске. Время доступа процессора к диску значительно превышает аналогичное время доступа к физической (оперативной) памяти. Чтобы процессор мог выполнять программы быстрее, они сначала помещаются в физическую (оперативную) память. После завершения своей работы, они возвращаются обратно на жесткий диск. Освобожденная таким образом память может быть выделена новой программе. При выполнении данные программы называются **процессами**.

**Виртуальная память** (или ***«логическая память»***) — это метод управления памятью, осуществляемый операционной системой, который позволяет программам задействовать значительно больше памяти, чем фактически установлено в компьютере. Например, если объем физической памяти компьютера составляет 4 ГБ, а виртуальной 16 ГБ, то программе может быть доступен объем виртуальной памяти вплоть до 16 ГБ.

| Физическая память | Виртуальная память |
|---|---|
| Непосредственно установленная в компьютере оперативная память. | Метод управления памятью, с помощью которого для программ создается иллюзия наличия в системе (физической) памяти, гораздо больше реально установленной. |
| Работает быстрее. | Работает медленнее. |
| Ограничена размером чипа ОЗУ. | Ограничена размером жесткого диска. |
| Может напрямую обращаться к процессору. | Не может напрямую обращаться к процессору. |
| Использует *swapping*. | Использует *paging*. |

**Физическая (оперативная) память** использует swapping. **Swapping** — это концепция управления памятью, при которой всякий раз, когда системе для хранения данных некоторого процесса не хватает оперативной (физической) памяти, она берет её из вторичного хранилища (например, жесткого диска), сбрасывая на него временно неиспользуемые

данные. В **Linux** есть специальная программа управления памятью, которая управляет этим процессом. Всякий раз, когда ОЗУ не хватает памяти, программа управления памятью ищет все те неактивные блоки данных (*страницы*), присутствующие в ОЗУ, которые не использовались в течение длительного времени. Когда она успешно находит подобные блоки, то перемещает их в память подкачки (например, на жесткий диск). Таким образом, освобождается пространство оперативной памяти, и, следовательно, его можно использовать для некоторых других программ, которые нуждаются в срочной обработке.

**Виртуальная память использует paging.** **Paging** — это метод выделения памяти, при котором разным несмежным блокам памяти назначается фиксированный размер. Размер обычно составляет 4 КБ. Paging всегда выполняется между активными *страницами* (*pages*).

**Методы для stress-ng –misaligned-method**

- 'all': Этот параметр запускает все типы невыровненных доступов к памяти.
- 'int16rd': Чтение 16-битных данных из невыровненных адресов.
- 'int16wr': Запись 16-битных данных в невыровненные адреса.
- 'int16inc': Инкремент 16-битных данных в невыровненных адресах.
- 'int16atomic': Атомарные операции над 16-битными данными в невыровненных адресах. 'int32rd': Чтение 32-битных данных из невыровненных адресов.
- 'int32wr': Запись 32-битных данных в невыровненные адреса.
- 'int32wrnt': Неоптимальная запись 32-битных данных в невыровненные адреса (несколько записей).
- 'int32inc': Инкремент 32-битных данных в невыровненных адресах.
- 'int32atomic': Атомарные операции над 32-битными данными в невыровненных адресах.
- 'int64rd': Чтение 64-битных данных из невыровненных адресов.
- 'int64wr': Запись 64-битных данных в невыровненные адреса.
- 'int64wrnt': Неоптимальная запись 64-битных данных в невыровненные адреса (несколько записей).
- 'int64inc': Инкремент 64-битных данных в невыровненных адресах.
- 'int64atomic': Атомарные операции над 64-битными данными в невыровненных адресах.
- 'int128rd': Чтение 128-битных данных из невыровненных адресов.
- 'int128wr': Запись 128-битных данных в невыровненные адреса.
- 'int128inc': Инкремент 128-битных данных в невыровненных адресах.
- 'int128atomic': Атомарные операции над 128-битными данными в невыровненных адресах.

**Атомарные операции в компьютерных науках — это операции, которые выполняются целиком или не выполняются вовсе, не допуская прерывания или вмешательства других операций.**

**Метрики sar -r**

1. *kbmemfree*: Количество свободной оперативной памяти в килобайтах (KB).
2. *kbavail*: Количество доступной оперативной памяти для процессов в килобайтах (KB).
3. *kbmemused*: Количество использованной оперативной памяти в килобайтах (KB).
4. *%memused*: Процент использования оперативной памяти от общего объёма.
5. *kbbuffers*: Количество оперативной памяти, используемой для буферизации дисковых блоков в килобайтах (KB).
6. *kbcached*: Количество оперативной памяти, используемой для кэширования данных из файловой системы в килобайтах (KB).
7. *kbcommit*: Общий размер оперативной памяти, зарезервированный для процессов на выполнение и для возможного использования подкачки (SWAP), в килобайтах (KB).
8. *%commit*: Процент коммита оперативной памяти от общего объёма.
9. *kbactive*: Количество оперативной памяти, активно используемой процессами в килобайтах (KB).
10. *kbinact*: Количество неактивной оперативной памяти, к которой не обращались длительное время в килобайтах (KB).
11. *kbdirty*: Количество оперативной памяти, содержащей данные, ожидающие запись на диск в килобайтах (KB).

**Стратегии управления данными**

1. **MADV_NORMAL:** Это значение по умолчанию. Нет особых требований к стратегии управления данными.
2. **MADV_RANDOM:** Предупреждает систему о случайном доступе к данным, чтобы она могла подстроиться и, возможно, кэшировать данные для случайного доступа.
3. **MADV_SEQUENTIAL:** Сообщает системе, что приложение будет последовательно обращаться к данным, и система может оптимизировать кэширование или загрузку страниц для последовательного доступа.
4. **MADV_WILLNEED:** Предупреждает о том, что приложению скоро потребуются данные из определенного диапазона виртуальной памяти, что позволяет системе начать предварительную загрузку этих данных.

5. **MADV_DONTNEED:** Сигнализирует о том, что приложение больше не будет использовать определенный диапазон памяти, и система может освободить физические страницы памяти, занимаемые этими данными.
6. **MADV_REMOVE:** Позволяет удалить данные страниц из файловой системы, предоставляя оптимизацию для операций чтения и записи на диске.
7. **MADV_FREE:** Помечает страницы памяти как свободные для использования ядром в любых целях.

## Блок 5. Network

Сетевые интерфейсы проводного интернета Ethernet обычно имеют имя, начинающиеся с символов **enp**, например, **enp3s0**. Такое именование используется только если ваш дистрибутив использует systemd, иначе будет применена старая система именования, при которой имена начинаются с символов **eth**, например **eth0**. Беспроводные сетевые интерфейсы, обычно называются **wlp** или **wlx** при использовании systemd, например, **wlp3s0**. Без использования systemd имя беспроводного интерфейса будет начинаться с **wlan**, например **wlan0**. Все остальные интерфейсы обычно виртуальные. Один из самых основных виртуальных интерфейсов - **lo**. Это локальный интерфейс, который позволяет программам обращаться к этому компьютеру.

Список можно получить:

**nmcli device status** – это я использовала

**ifconfig** Метрики

1. **enp0s3** - это название сетевого интерфейса.
    - **flags=4163** - флаги, указывающие на состояние интерфейса. Например, **UP** (включен), **BROADCAST** (поддерживает широковещательные сообщения), **RUNNING** (интерфейс запущен), **MULTICAST** (поддерживает мультикаст).
    - **mtu 1500** - максимальный размер пакета (Maximum Transmission Unit) в байтах.

Затем идет информация о сетевых адресах:

- **inet 10.0.2.15** - IPv4 адрес.
- **netmask 255.255.255.0** - сетевая маска IPv4 адреса.
- **broadcast 10.0.2.255** - широковещательный адрес для сети.
- **inet6 fe80::1ac2:f148:e746:90f4** - IPv6 адрес.

- **prefixlen 64** - длина префикса IPv6 адреса.
- **scopeid 0x20<link>** - область применения адреса (link-local).

Затем идет информация о физическом интерфейсе:

- **ether 08:00:27:2a:69:2c** - MAC-адрес (физический адрес) интерфейса.
- **txqueuelen 1000** - длина очереди передачи пакетов Ethernet.

И, наконец, статистика использования сети:

- **RX packets 7702** - количество принятых пакетов.
- **bytes 11155344** - количество принятых байт.
- **TX packets 1805** - количество переданных пакетов.
- **bytes 134299** - количество переданных байт.
- **RX errors 0** - количество ошибок при приеме.
- **TX errors 0** - количество ошибок при передаче.
- **collisions 0** - количество коллизий в сети.

**Iptraf** general

- Iface - название сетевого интерфейса.
- Total - общее количество переданных или принятых пакетов.
- IPv4 - количество IPv4-пакетов.
- IPv6 - количество IPv6-пакетов.
- NonIP - количество пакетов, не относящихся к IP-адресации (например, ARP или другие протоколы на более низком уровне).
- BadIP - количество некорректных пакетов, которые не удалось обработать или идентифицировать.
- Activity - скорость трафика в килобитах в секунду (kbps).

**Iptraf** detailed stat

- *Total Packets* и *Total Bytes*: Общее количество переданных и принятых пакетов и объем данных в байтах.
- *Incoming Packets* и *Incoming Bytes*: Количество и объем данных во входящих пакетах (пришедших на интерфейс).

- *Outgoing Packets* и *Outgoing Bytes*: Количество и объем данных в исходящих пакетах (уходящих с интерфейса).
- *IPv4* и *IPv6*: Количество и объем данных для IPv4 и IPv6 пакетов соответственно.
- *TCP, UDP, ICMP, Other IP*: Количество и объем данных для соответствующих протоколов.
- *Non-IP*: Количество и объем данных для пакетов, не относящихся к IP (например, ARP и другие протоколы на более низком уровне).
- *Broadcast*: Количество и объем данных в широковещательных пакетах.
- *Total rates, Incoming rates, Outgoing rates*: Скорость передачи данных в килобитах в секунду (kbps) и количество пакетов в секунду (pps) в общем, входящем и исходящем направлениях.
- *IP checksum errors*: Количество обнаруженных ошибок контрольной суммы IP.

**Sar -n SOCK** метрики

- **totsck**: Это общее количество сокетов (коммуникационных конечных точек), которые в настоящее время открыты на вашей системе. Сокеты могут использоваться для установки соединений, передачи данных по сети и других сетевых операций.
- **tcpsck**: Это количество сокетов TCP (Transmission Control Protocol), которые в настоящее время открыты. TCP используется для установления надежных соединений между устройствами в сети и гарантирует доставку данных в правильной последовательности и без потерь.
- **udpsck**: Это количество сокетов UDP (User Datagram Protocol), открытых в данный момент времени. UDP предоставляет более простой способ передачи данных, но без гарантии доставки или порядка. Это часто используется для более быстрой передачи данных или там, где потеря пакетов не является критической проблемой.
- **rawsck**: Это количество RAW сокетов, которые в настоящее время открыты. RAW сокеты предоставляют доступ к сырым сетевым протоколам, позволяя приложениям более прямо работать с пакетами данных на более низком уровне, чем TCP или UDP.
- **ip-frag**: Это количество сокетов, используемых для обработки фрагментации IP пакетов. Фрагментация происходит, когда пакет

данных слишком большой для передачи по сети и разделяется на несколько меньших фрагментов.

- **tcp-tw**: Это количество TCP сокетов в состоянии TIME-WAIT. TIME-WAIT - это состояние, в котором находятся сокеты TCP после завершения соединения, чтобы убедиться, что все пакеты из предыдущего соединения завершили свой путь по сети, прежде чем сокет будет окончательно закрыт.

**Netlink Taskstats** - это механизм в ядре Linux, который предоставляет информацию о работе процессов. Он позволяет пользовательским программам получать детализированную статистику о процессах в системе. Эта информация включает в себя различные метрики, такие как количество времени CPU, потребление памяти, количество созданных и завершенных процессов, статусы процессов и многое другое.

Как работает Netlink Taskstats:

1. **Сбор данных**: При выполнении каждого процесса ядро Linux собирает статистику о его выполнении, такую как использование CPU, использование памяти, количество системных вызовов и т.д.

2. **Доступ к данным**: Эта статистика доступна через механизм Netlink. Пользовательские программы могут запросить эти данные через Netlink API для мониторинга процессов и анализа их работы.

3. **Интерпретация данных**: Полученные данные могут быть проанализированы пользовательскими программами для мониторинга и оптимизации работы процессов. Например, утилиты мониторинга, такие как `sar`, `top` и другие, используют эти данные для отображения статистики процессов и общей активности системы.

Данные Taskstats могут быть полезны при анализе производительности, выявлении проблем в работе программ, оптимизации алгоритмов и тестировании программного обеспечения. Эти метрики могут также помочь в выявлении и устранении узких мест в производительности приложений или системы в целом.

## Блок 6. Pipe

**Pipe** (конвеер) – это однонаправленный канал межпроцессного взаимодействия.

## Блок 7. Sched

Scheduler – планировщик задач

Планирование – это процесс распределения ресурсов системы для выполнения задач. В статье мы рассмотрим его вариант, в котором ресурсом является одно или несколько ядер процессора, а задачи представлены потоками или процессами, которые нужно выполнить.

В Linux процессы делятся на два типа:

- Процессы реального времени.
- Условные процессы.

Процессы реального времени должны вписываться в границы времени ответа, независимо от загрузки системы. Иначе говоря, такие процессы являются срочными и ни при каких условиях не откладываются.

В качестве примера можно привести процесс переноса, отвечающий за распределение рабочей нагрузки между ядрами ЦПУ.

Условные же процессы не ограничиваются строгими рамками времени ответа и в случае занятости системы могут подвергаться задержкам.

В пример можно привести процесс браузера, который позволяет читать статьи.

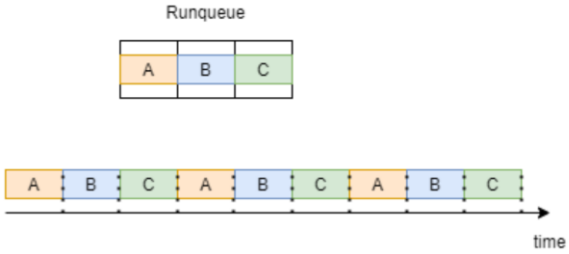
Алгоритмы планировщика

**SCHED_FIFO**
В данной политике планировщик выбирает процесс, ориентируясь на время его поступления (*FIFO = первым вошел, первым вышел*).

Процесс с политикой планирования SCHED_FIFO может «освободить» ЦПУ в нескольких случаях:

- Процесс ожидает, к примеру, операции ввода/вывода, после чего по возвращению в состояние «готов» помещается в конец очереди.
- Процесс уступил ЦПУ через системный вызов sched_yield, после чего он тут же возвращается в конец очереди.

**SCHED_RR**
SCHED_RR подразумевает циклическое планирование.
В этой политике каждый процесс в очереди получает интервал времени (квант) и выполняется в свою очередь (исходя из приоритета) по циклическому принципу.
Для лучшего понимания рассмотрим пример, где в очереди находятся три процесса, A B C, все из которых работают по политике SCHED_RR.
Как показано ниже, каждый процесс получает квант времени и выполняется в свою очередь. После однократного выполнения всех процессов они повторяются в той же последовательности.




**CFS**—Абсолютно справедливый планировщик
*CFS* задействует *красно-черное дерево*, представляющее бинарное дерево поиска – то есть добавление, удаление и поиск выполняются за O(logN), где N выражает количество процессов.

Ключом в этом дереве выступает виртуальное время выполнения процесса. Новые процессы или процесс, возвращающиеся из ожидания в состояние готовности, добавляются в дерево с ключом vruntime = min_vruntime. Это очень важный момент, который позволяет избежать дефицита внимания ЦПУ для старых процессов.
В первую очередь алгоритм устанавливает себе *лимит времени – sched_latency*.
В течение этого времени алгоритм стремится выполнить все готовые процессы – N. Это означает, что каждый процесс получит интервал времени равный временному лимиту, поделенному на количество процессов: *Qi = sched_latency/N.*
Когда процесс исчерпывает свой интервал (Qi), алгоритм выбирает в дереве следующий процесс с наименьшим виртуальным временем.

| **RR – циклический список** | **CFS – абсолютно справедливый планировщик** |
|---|---|
| • Квант времени статичен и не зависит от количества процессов в системе. | • Квант времени динамичен и может изменяться в соответствии с количеством процессов в системе. |

| | |
|---|---|
| • По исчерпанию процессом его кванта времени, RR выбирает очередной процесс с наименьшим виртуальным временем из циклического списка. | • По исчерпанию процессом его кванта времени, CFS выбирает очередной процесс с наименьшим виртуальным временем из красно-черного дерева. |

**Iostat метрики:**

1. **avg-cpu**:

- `%user`: Процент времени CPU, затраченный на выполнение пользовательских процессов.

- `%nice`: Процент времени CPU, затраченный на выполнение процессов с приоритетом "nice".

- `%system`: Процент времени CPU, затраченный на выполнение системных процессов ядра.

- `%iowait`: Процент времени, в течение которого процессор ожидал завершения операций ввода-вывода.

- `%steal`: Процент времени, когда процессор был украден (виртуализированным) другими виртуальными машинами на физическом хосте.

- `%idle`: Процент времени, когда процессор был бездействующим (не выполнял никаких задач).

2. **Device**:

- `tps`: Количество операций ввода-вывода (транзакций в секунду) на устройстве.

- `kB_read/s`: Количество килобайт данных, считанных с устройства в секунду.

- `kB_wrtn/s`: Количество килобайт данных, записанных на устройство в секунду.

- `kB_dscd/s`: Количество килобайт данных, сброшенных с устройства в секунду (например, для устройств типа loop это может быть нулевым).

3. **Сводные данные для каждого устройства**:

- Имя устройства (например, `sda`, `loop0`, `sr1` и т.д.).

- `tps`: Количество операций ввода-вывода в секунду.

- `kB_read/s`: Количество килобайт, считанных с устройства в секунду.

- `kB_wrtn/s`: Количество килобайт, записанных на устройство в секунду.

- `kB_read`: Общее количество считанных килобайт.

- `kB_wrtn`: Общее количество записанных килобайт.

- `kB_dscd`: Общее количество сброшенных килобайт (например, для некоторых устройств это может быть нулевым).

Эти метрики помогают в мониторинге активности устройств ввода-вывода и позволяют определить, сколько данных считывается и записывается на каждом устройстве в режиме реального времени.

---

Опция `--hdd` в утилите `stress-ng` предназначена для создания нагрузки на дисковую подсистему системы. При использовании этой опции `stress-ng` генерирует интенсивную нагрузку на диск, создавая операции чтения и записи на диск.

Аргумент `--hdd-bytes 1g` указывает, сколько данных нужно использовать в процессе нагрузки на диск. В данном случае `1g` означает 1 гигабайт данных. Это означает, что `stress-ng` будет создавать операции чтения и записи на диск для генерации 1 гигабайта данных, создавая интенсивную дисковую активность размером 1 гигабайт.

## Описание команд из блока 1.9 Лекции 1

## Sar

**-B –** мониторинг пэйджинга

Основная идея страничного управления заключается в том, чтобы разбить физическую память и виртуальную память программы на небольшие кусочки (страницы). Когда программа запрашивает доступ к памяти, операционная система работает с этими страницами, переносит их между оперативной памятью и хранилищем (например, жестким диском) при необходимости.

Paging помогает обеспечить следующие важные функции:

1. **Виртуальная память**: Позволяет приложениям использовать больше памяти, чем физически доступно. ОС управляет памятью, используя комбинацию физической памяти и файлов на диске в качестве расширения оперативной памяти.

2. **Управление памятью**: Операционная система использует страничное управление для эффективного выделения, освобождения и распределения памяти между различными процессами и задачами.
3. **Контроль доступа**: Путем использования страниц и различных уровней доступа к ним (например, через атрибуты защиты) операционная система обеспечивает безопасность данных и предотвращает несанкционированный доступ к памяти.
4. **Улучшение производительности**: Помогает оптимизировать использование памяти, позволяя операционной системе эффективно управлять памятью и минимизировать время доступа к данным, которые временно не используются программами.

*Метрики*

- *pgpgin/s:* Среднее количество страниц, прочитанных из блочных устройств в секунду. Отображает количество страниц, скопированных из блочных устройств (как правило, жесткого диска) в оперативную память в секунду. Это обычно происходит при обработке страниц, которые ранее были выгружены на диск, и возвращении их в оперативную память в результате запроса от программ.
- *pgpgout/s:* Среднее количество страниц, записанных в блочные устройства в секунду. Отражает количество страниц, которые были выгружены из оперативной памяти на блочные устройства в секунду. Это может произойти, например, при выгрузке неиспользуемых страниц памяти на жесткий диск для освобождения места.
- *fault/s*: Среднее количество мажорных и минорных ошибок в секунду. Это общее количество ошибок памяти в секунду. Минорные ошибки происходят, когда страницы памяти отсутствуют в кэше и должны быть загружены из диска. Мажорные ошибки обозначают ошибки, которые требуют чтения данных с диска, такие как страницы, которые были физически удалены из памяти и сейчас необходимо восстановить.
- *majflt/s*: Среднее количество мажорных ошибок в секунду. Это количество мажорных ошибок (ошибок, требующих обращения к блочному устройству) в секунду.
- *pgfree/s:* Среднее количество страниц, освобожденных в секунду. Отображает количество страниц памяти, которые были освобождены в результате действий, таких как освобождение кеша страниц или страниц, которые не используются.
- *pgscank/s:* Среднее количество страниц, сканированных в кеше в секунду.
- *pgscand/s:* Среднее количество сканированных страниц в секунду.
- *pgsteal/s:* Среднее количество страниц, вытесненных из кеша в секунду.

- *%vmeff*: Процент эффективности виртуальной памяти. Это отношение общего числа страниц, вошедших и вышедших, к общему числу страниц в системе. Этот показатель может служить оценкой эффективности использования виртуальной памяти.

**-W –** мониторинг своппинга

**Swapping** (или ***«подкачка памяти»***) — это процесс, при котором страница памяти копируется в специальное пространство на жестком диске, называемое **пространством подкачки** (англ. ***«swap space»***), освобождая занимаемый ею объем оперативной памяти. В Linux для этого существует специальная программа — менеджер памяти. Каждый раз, когда системе требуется больше памяти, чем ей физически доступно в данный момент, менеджер памяти ищет все редко используемые страницы памяти и вытесняет их на жесткий диск, предоставляя освободившуюся память другому приложению (процессу).

Метрики

- **pswpin/s** отражает среднее количество процессов, которые были помещены в область подкачки (swap) за секунду. Это число указывает на то, как часто операционная система вынуждена перемещать данные из оперативной памяти на жесткий диск.
- **pswpout/s** показывает среднее количество процессов, которые были извлечены из области подкачки за секунду. Это значение указывает на частоту возврата данных из файла подкачки в оперативную память.

**-b** - отчет о статистике ввода-вывода (I/O) и скорости передачи данных.

- *Tps* - Общее количество операций ввода-вывода в секунду, которые были отправлены на физические устройства. Операция передачи данных - это запрос на ввод-вывод к физическому устройству. Несколько логических запросов могут быть объединены в один запрос на ввод-вывод к устройству. Размер передачи неопределен.

- *Rtps* - Общее количество запросов на чтение в секунду, отправленных на физические устройства.

- *Wtps* - Общее количество запросов на запись в секунду, отправленных на физические устройства.

- *bread/s* - Общий объем данных, прочитанных с устройств блоками в секунду. Блоки эквивалентны секторам и, следовательно, имеют размер 512 байт.

- *bwrtn/s* - Общий объем данных, записанных на устройства блоками в секунду.

**-d** Отчет о активности для каждого блочного устройства. Когда отображаются данные, обычно используется спецификация устройства dev m-n (столбец DEV). m - это основной номер устройства, а n - его дополнительный номер. Имена устройств также могут быть представлены в удобном формате, если используется параметр -p, или можно вывести постоянные имена устройств с помощью параметра -j (см. ниже). Обратите внимание, что активность диска зависит от опций sadc -S DISK и -S XDISK для сбора данных.

- *Tps* - Количество передач в секунду, отправленных на устройство. Несколько логических запросов могут быть объединены в один запрос на ввод-вывод к устройству. Размер передачи неопределен.

- *rd_sec/s* - Количество секторов, прочитанных с устройства. Размер сектора составляет 512 байт.

- *wr_sec/s* - Количество секторов, записанных на устройство. Размер сектора составляет 512 байт.

- *avgrq-sz* - Средний размер (в секторах) запросов, отправленных на устройство.

- *avgqu-sz* - Средняя длина очереди запросов, отправленных на устройство.

- *Await* - Среднее время (в миллисекундах) ожидания обслуживания запросов на ввод-вывод к устройству. Это включает время ожидания запросов в очереди и время обслуживания.

- *Svctm* - Среднее время обслуживания (в миллисекундах) запросов на ввод-вывод к устройству. Внимание! Не рекомендуется больше полагаться на это поле. Оно будет удалено в будущих версиях sysstat.

- *%util* - Процент времени процессора, в течение которого запросы на ввод-вывод были отправлены на устройство (использование

пропускной способности устройства). Насыщение устройства происходит, когда это значение близко к 100%.

**-F -** Этот отчет выводит статистику для в настоящее время подключенных файловых систем, игнорируя псевдо-файловые системы. По завершении отчета, sar отобразит сводку по всем этим файловым системам. Статистика файловых систем зависит от опции сбора данных -S XDISK для sadc. Ниже перечислены отображаемые значения:

- **MBfsfree**: Общий объем свободного пространства в мегабайтах (включая пространство, доступное только для привилегированного пользователя).

- **MBfsused**: Общий объем используемого пространства в мегабайтах.

- **%fsused**: Процент использованного пространства файловой системы, видимый привилегированным пользователем.

- **%ufsused**: Процент использованного пространства файловой системы, видимый непривилегированным пользователем.

- **Ifree**: Общее количество свободных файловых узлов в файловой системе.

- **Iused**: Общее количество используемых файловых узлов в файловой системе.

- **%Iused**: Процент использования файловых узлов в файловой системе.

**-I** { int [,...] | SUM | ALL | XALL } позволяет отобразить статистику для определенного прерывания (интеррапта). Вот перевод значений, которые можно указать в этой опции:

- `int` - номер интеррапта. Указание нескольких значений `int` через запятую позволит просматривать статистику для нескольких независимых интерраптов.

- `SUM` - ключевое слово, указывающее, что нужно отобразить общее количество прерываний, полученных в секунду.

- `ALL` - ключевое слово, указывающее, что нужно отобразить статистику для первых 16 прерываний.

- `XALL` - ключевое слово, указывающее, что нужно отобразить статистику для всех прерываний, включая потенциальные источники прерываний APIC (Advanced Programmable Interrupt Controller).

Обратите внимание, что сбор статистики прерываний зависит от опции сбора данных `sadc -S INT`.

**-y** Отчет о активности устройств TTY (терминалов) отображает следующие значения:

- `rcvin/s` - Количество прерываний на прием данных в секунду для текущей последовательной линии (серийного порта). Номер последовательной линии указан в столбце TTY.

- `xmtin/s` - Количество прерываний на передачу данных в секунду для текущей последовательной линии.

- `framerr/s` - Количество ошибок кадра (frame) в секунду для текущей последовательной линии.

- `prtyerr/s` - Количество ошибок четности (parity) в секунду для текущей последовательной линии.

- `brk/s` - Количество обнаруженных разрывов (breaks) в секунду для текущей последовательной линии.

- `ovrun/s` - Количество ошибок переполнения буфера (overrun) в секунду для текущей последовательной линии.

TTY (Teletypewriter) - это термин, который оригинально обозначал физическое устройство, используемое для ввода и вывода текста в компьютере или системе. Сейчас этот термин часто используется для обозначения виртуальных консолей в Unix-подобных операционных системах.

**-q -** Отчет о длине очереди и средних нагрузках. Отображаются следующие значения:

- runq-sz - Длина очереди исполнения (количество задач, ожидающих времени исполнения).
- plist-sz - Количество задач в списке задач.
- ldavg-1 - Средняя системная загрузка за последнюю минуту. Рассчитывается как среднее количество выполнимых или выполняемых задач (в состоянии R) и количество задач в состоянии непрерываемого сна (D state) за указанный интервал.
- ldavg-5 - Системная средняя нагрузка за последние 5 минут.
- ldavg-15 - Системная средняя нагрузка за последние 15 минут.
- Blocked - Количество задач в настоящее время заблокированных, ожидающих завершения операций ввода-вывода.

**-P -** Отчет о статистике для каждого процессора или указанных процессоров. Указание ключевого слова ALL позволяет получить статистику для каждого отдельного процессора, а также глобально для всех процессоров. Обратите внимание, что процессор 0 является первым процессором.

**-** ‘CPU’ – номер процессора (all, 0, 1, 2, 3)

- `%user`: Процент времени CPU, затраченный на выполнение пользовательских процессов.

- `%nice`: Процент времени CPU, затраченный на выполнение процессов с приоритетом "nice".

- `%system`: Процент времени CPU, затраченный на выполнение системных процессов ядра.

- `%iowait`: Процент времени, в течение которого процессор ожидал завершения операций ввода-вывода.

- `%steal`: Процент времени, когда процессор был украден (виртуализированным) другими виртуальными машинами на физическом хосте.

- `%idle`: Процент времени, когда процессор был бездействующим (не выполнял никаких задач).

**Strace**

1. `execve("/usr/bin/ls", ["ls"], 0x7ffeca268440 /* 54 vars */)`: Запускается команда `ls` из директории `/usr/bin`.

2. `brk(NULL) = 0x55dd4a616000`: Вызов системного вызова `brk`, который устанавливает верхнюю границу кучи процесса (память для динамического выделения) до указанного адреса.

3. `arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd860233e0) = -1 EINVAL (Invalid argument)`: Попытка установить архитектурные параметры процессора.

4. `mmap(...)`: Здесь идут вызовы `mmap`, которые отображают файлы библиотек (`libselinux.so.1`, `libc.so.6`, `libpcre2-8.so.0`) в виртуальное адресное пространство процесса.

5. `munmap(0x7f42a1133000, 61603) = 0`: Это вызов `munmap`, который удаляет отображение памяти (в данном случае, отображение содержимого `ld.so.cache`).

6. `statfs()`: Этот вызов используется для получения информации о файловой системе.

7. `getrandom("\xd3\xb5\xe9\x37\x2c\x16\xcb\xa0", 8, GRND_NONBLOCK) =8`: Получение случайных данных.

8. `ioctl()`: Вызовы `ioctl` используются для управления устройствами ввода/вывода.

9. `write()`: Запись в стандартный вывод.

10. `exit_group(0) = ?`: Завершение процесса с кодом возврата 0.

11. Таким образом, вывод **access("/etc/ld.so.preload", R_OK) = -1 ENOENT** говорит о том, что программа пыталась проверить доступ к файлу **/etc/ld.so.preload**, но этот файл не был найден в системе. Файл **/etc/ld.so.preload** - это конфигурационный файл в системе Linux, который может быть использован для загрузки предварительно определенных библиотек перед выполнением программ. В этом файле указываются пути к различным библиотекам, которые будут загружены в память при старте программ. (у Клименкова тоже так)

12. close(n) – закрытие файлового дескриптора n

13. **newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=166280, ...}, AT_EMPTY_PATH) = 0**: Этот вызов **newfstatat** с аргументами 3 (файловый дескриптор), пустой строкой (путь) и флагом **AT_EMPTY_PATH** возвращает информацию о файле с дескриптором 3, который здесь является файлом библиотеки. Этот вызов является улучшенной версией стандартного вызова **fstatat**.

14. **mprotect** используется для изменения атрибутов страниц памяти. В данном случае, область памяти с адреса 0x7f42a110d000 длиной 139264 байт помечается как недоступная для записи и выполнения (**PROT_NONE**).

Это пример типичного вывода `strace` для команды `ls`, который показывает системные вызовы, используемые при выполнении этой команды.

**Это для strace ls**

Это файлы библиотек, используемые программой `ls`. Вот краткое описание каждой из них:

1. **libselinux.so.1**: Это библиотека, относящаяся к Security-Enhanced Linux (SELinux). SELinux - это механизм обеспечения безопасности в ядре Linux, который предоставляет расширенные механизмы контроля доступа для усиления безопасности операционной системы.

2. **libc.so.6**: Это библиотека стандартной библиотеки языка программирования Си. `libc` (C библиотека) содержит функции, которые используются практически всеми программами на Linux для взаимодействия с ядром и выполнения основных операций ввода-вывода, аллокации памяти, работы со строками и т. д.

3. **libpcre2-8.so.0**: Это библиотека Perl Compatible Regular Expressions (PCRE) версии 2. PCRE - это библиотека для работы с регулярными выражениями в стиле Perl, используемая для поиска и обработки строк по шаблону, предоставляющая средства для работы с текстом на основе определенных правил.

Эти библиотеки являются основными и используются множеством программ в системе Linux для обеспечения базовой функциональности, работы с безопасностью (в случае SELinux) и поддержки регулярных выражений.

Файловый дескриптор (file descriptor) в операционных системах Unix-подобных системах - это неотрицательное целое число, которое идентифицирует открытый файл, устройство или другой объект, с которым процесс взаимодействует через операции ввода-вывода.

ОС Unix представляет каждый открытый файл (или устройство) как файловый дескриптор. Это включает в себя обычные файлы, директории, сетевые сокеты, каналы FIFO и другие объекты.

Когда процесс открывает файл, ему выделяется файловый дескриптор, который используется для всех последующих операций ввода-вывода с этим файлом. Например, чтение, запись, перемещение указателя в файле, закрытие и т. д.

Стандартные дескрипторы, связанные с каждым процессом:

- **0 (stdin)**: Стандартный ввод (обычно клавиатура).

- **1 (stdout)**: Стандартный вывод (обычно экран или консоль).

- **2 (stderr)**: Стандартный вывод ошибок (также обычно экран или консоль).

Для манипуляции с файловыми дескрипторами используются различные системные вызовы, такие как `open`, `close`, `read`, `write`, `fcntl` и другие.

Файловые дескрипторы также могут быть использованы для взаимодействия с сетевыми ресурсами, такими как сетевые сокеты, а также для межпроцессного взаимодействия через каналы или сегменты разделяемой памяти.

**Ltrace (-fS)**

По факту выводит вызовы библиотек из strace -f -несистемные библотеки, -S – системные

**Bpftrace**

bpftrace -l 'tracepoint:syscalls:sys_enter_*'

"bpftrace -l" выводит все точки инструментирования (probe), к которым можно добавить поисковый запрос.

Точка инструментирования (probe) - это место инструментации для захвата данных событий.

Поддерживаемый поисковый запрос поддерживает метасимволы/шаблоны (* и ?).

Команда "bpftrace -l" также может быть направлена через конвейер на grep(1) для полного поиска с использованием регулярных выражений.

**bpftrace** - это инструмент трассировки в системе Linux, который использует eBPF (extended Berkeley Packet Filter) для создания динамических и мощных программ трассировки.

Он предоставляет простой в использовании язык для написания скриптов, который позволяет анализировать различные аспекты работы системы в реальном времени, такие как системные вызовы, сетевая активность, использование ресурсов CPU и многое другое. Это позволяет инженерам систем, разработчикам и администраторам проводить мониторинг и анализ системы в реальном времени без значительных нагрузок на саму систему.

**bpftrace** позволяет создавать скрипты для трассировки, состоящие из инструкций, аналогичных языку программирования C. Он предоставляет API для обращения к множеству системных событий и данных, обеспечивая возможность углубленного анализа системы и её поведения.

Например, вы можете использовать **bpftrace** для:

- Отслеживания системных вызовов и их аргументов.

- Мониторинга использования CPU и других ресурсов системы.
- Отслеживания сетевой активности.
- Профилирования работы программ.
- Исследования работы ядра Linux и его подсистем.

Использование **bpftrace** требует некоторых привилегий доступа к системе, и в зависимости от трассируемых событий и данных, которые вы хотите анализировать, может потребоваться более высокий уровень привилегий или прав доступа.

sudo bpftrace -e 'tracepoint:raw_syscalls:sys_enter { @[comm] = count(); }'

Это сводка системных вызовов по имени процесса, выводящая отчет при нажатии Ctrl-C.

@: Это обозначает особый тип переменной, называемой "map" (карта), которая может хранить и подводить итоги данных различными способами. Можно добавить необязательное имя переменной после символа @, например, "@num", как для улучшения читаемости, так и для различения между более чем одной картой.

[]: Необязательные скобки позволяют установить ключ для карты, подобно ассоциативному массиву.

count(): Это функция карты – способ, которым она заполняется. count() подсчитывает количество раз, когда она вызывается. Поскольку эта информация сохраняется по comm (имени команды), результатом является частотный подсчет системных вызовов по имени процесса.

sudo perf stat -e cache-references,cache-misses stress-ng --cache 1 --cache-ways {i} --timeout 20