

Обычно процессы появляются, делают свою работу и завершаются не требуя никакого внимания. Но бывают случаи, когда администратору все же следует вмешаться. Давайте разберём пару таких ситуаций. Самая популярная – зависание процесса или системы. Я думаю все сталкивались с такими ситуациями, когда какой-то процесс начинает использовать слишком много оперативной памяти или ресурсов процессора. Это приводит к тому, что каким-то другим процессам, той же оболочке, перестаёт хватать ресурсов и всё начинает зависать. Чтобы решить проблему, сначала нужно понять, какой процесс во всём виноват. Для этого нужно увидеть список процессов, которые используют большую часть ресурсов. Это нам покажет утилита `top`.

```
user@centos8:~  
File Edit View Search Terminal Help  
top - 18:59:13 up 1 min, 1 user, load average: 1.34, 0.43, 0.15  
Tasks: 250 total, 3 running, 246 sleeping, 0 stopped, 1 zombie  
%Cpu(s): 0.4 us, 0.4 sy, 0.0 ni, 98.6 id, 0.4 wa, 0.4 hi, 0.0 si, 0.0 st  
MiB Mem : 1829.6 total, 170.5 free, 1144.1 used, 515.0 buff/cache  
MiB Swap: 2048.0 total, 2032.0 free, 16.0 used. 512.1 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3336	root	29	9	187536	8264	3700	R	2.0	0.4	0:00.06	(coredump)
2588	user	20	0	2875352	301744	111764	S	1.3	16.1	0:03.80	gnome-she+
1	root	20	0	187532	13344	8804	S	0.7	0.7	0:00.95	systemd
973	root	16	-4	143088	2712	2100	S	0.3	0.1	0:00.01	auditd
1024	polkitd	20	0	1635812	25924	18528	S	0.3	1.4	0:02.24	polkitd
1032	dbus	20	0	72328	7060	4148	S	0.3	0.4	0:00.46	dbus-daem+
3206	user	20	0	736524	46976	36552	S	0.3	2.5	0:00.23	gnome-ter+
3304	user	20	0	273768	4888	4036	R	0.3	0.3	0:00.09	top

Наверху отображается общая информация по системе. Давайте пройдемся по каждому из значений. Первая строчка – вывод утилиты `uptime`:

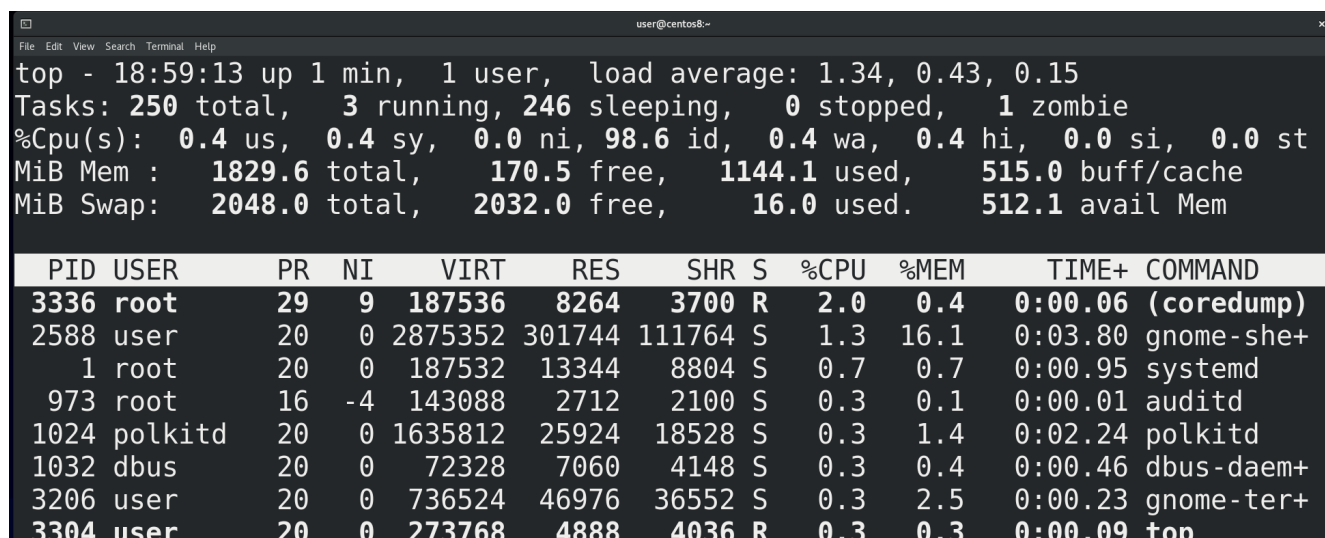
1. Текущее время в системе
2. Сколько времени система включена
3. Хотя тут и написано `user`, подразумевается количество залогиненных сессий. Это могут быть как сессии одного пользователя, так и других.

```
user@centos8:~  
File Edit View Search Terminal Help  
top - 19:04:38 up 6 min, 2 users, load average: 0.11, 0.21, 0.13  
Tasks: 242 total, 1 running, 241 sleeping, 0 stopped, 0 zombie
```

```
user@centos8 ~]$ w  
19:04:24 up 6 min, 2 users, load average: 0.14, 0.22, 0.13  
USER      TTY      FROM          LOGIN@      IDLE        JCPU        PCPU WHAT  
user      tty2     tty2          18:58       6:13       18.26s     0.77s /usr/bin/gno  
user      tty3     -             19:03      32.00s     0.24s     0.23s top  
[user@centos8 ~]$
```

Эмулятор терминала даёт нам оболочку без входа, т.е. в ней мы не логинимся, а вот виртуальный терминал – оболочку со входом. Давайте, для примера, я залогинюсь тем же пользователем в виртуальной терминале (`ctrl+alt+f3`). Теперь здесь отображается 2 пользователя. Кстати, информацию о залогиненных пользователях можно увидеть с помощью утилиты `w`.

4. Среднее значение загрузки системы за минуту, 5 минут и 15 минут. Есть свои нюансы подсчёта этого значения, рекомендую почитать [статью](#) на википедии.

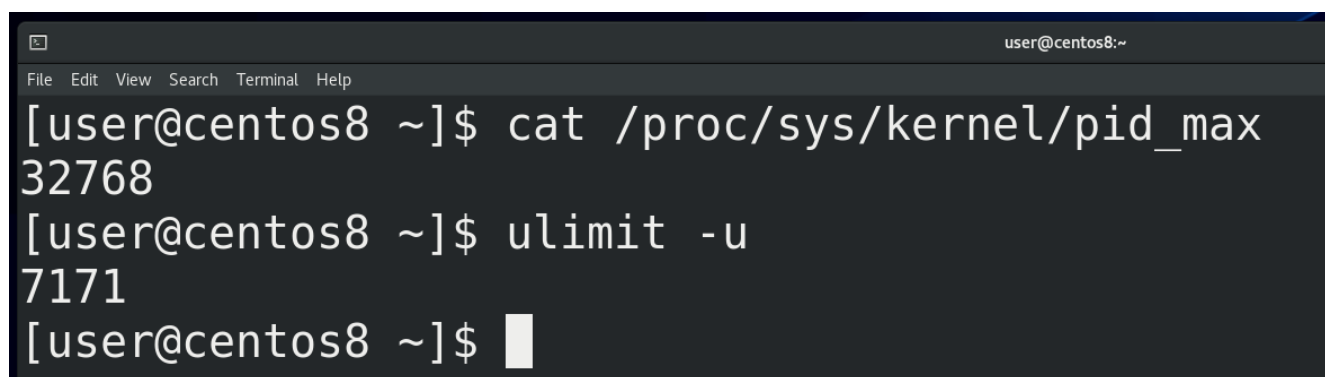


```
top - 18:59:13 up 1 min, 1 user, load average: 1.34, 0.43, 0.15
Tasks: 250 total, 3 running, 246 sleeping, 0 stopped, 1 zombie
%Cpu(s): 0.4 us, 0.4 sy, 0.0 ni, 98.6 id, 0.4 wa, 0.4 hi, 0.0 si, 0.0 st
MiB Mem : 1829.6 total, 170.5 free, 1144.1 used, 515.0 buff/cache
MiB Swap: 2048.0 total, 2032.0 free, 16.0 used. 512.1 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3336	root	29	9	187536	8264	3700	R	2.0	0.4	0:00.06	(coredump)
2588	user	20	0	2875352	301744	111764	S	1.3	16.1	0:03.80	gnome-she+
1	root	20	0	187532	13344	8804	S	0.7	0.7	0:00.95	systemd
973	root	16	-4	143088	2712	2100	S	0.3	0.1	0:00.01	auditd
1024	polkitd	20	0	1635812	25924	18528	S	0.3	1.4	0:02.24	polkitd
1032	dbus	20	0	72328	7060	4148	S	0.3	0.4	0:00.46	dbus-daem+
3206	user	20	0	736524	46976	36552	S	0.3	2.5	0:00.23	gnome-ter+
3304	user	20	0	273768	4888	4036	R	0.3	0.3	0:00.09	top

На второй строчке информация о количестве процессов:

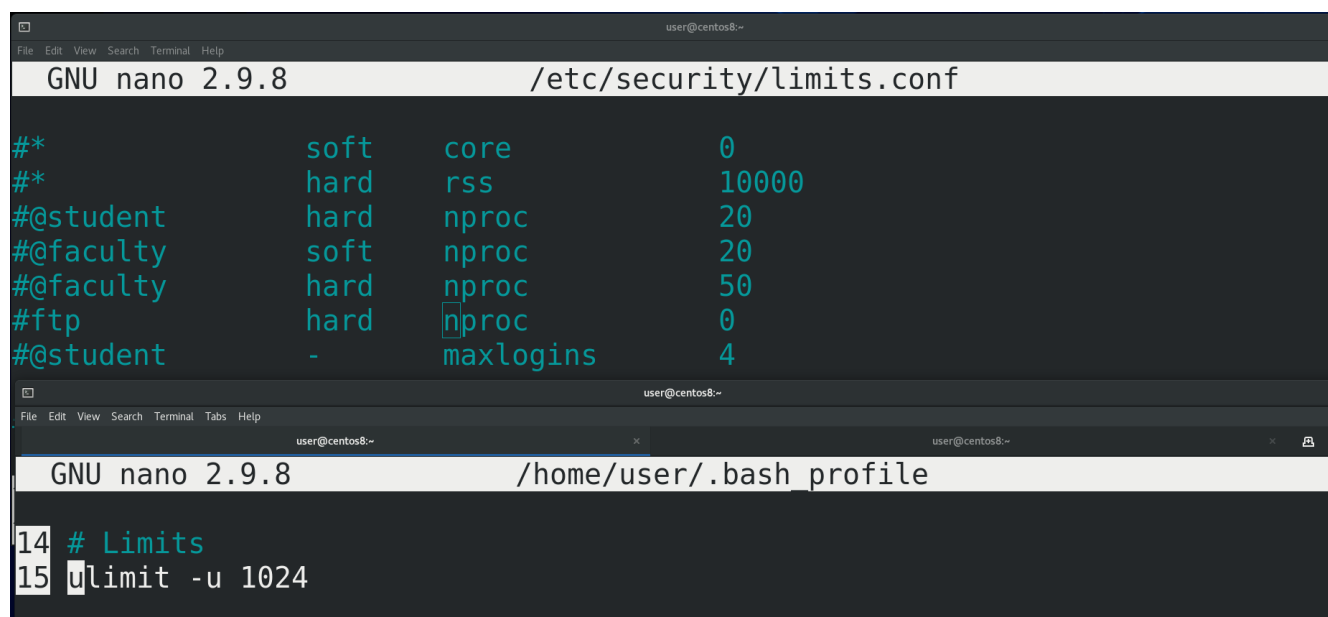
1. Сколько всего запущенных процессов
2. Сколько процессов выполняется в данный момент
3. Сколько процессов просто спят, например, в ожидании каких-то данных
4. Сколько процессов остановлены. Вообще, процессы можно останавливать. В этот момент они всё ещё находятся в оперативке, но перестают посылать код на процессор – просто ничего не делают. Допустим, когда несколько программ пишут данные на диск, то у диска перестаёт хватать скорости. Вы можете временно остановить какие-то из программ, чтобы другие ускорили свою работу. Но нужно понимать, что это очень специфично, и разные программы могут по разному на такое реагировать. Допустим, если программа работает с сетью и её остановить, то скорее всего сетевое соединение прервётся. Как останавливать программы мы разберём чуть позже.



```
[user@centos8 ~]$ cat /proc/sys/kernel/pid_max
32768
[user@centos8 ~]$ ulimit -u
7171
[user@centos8 ~]$
```

5. Сколько зомби процессов. Когда один процесс запускает другой, то он является для него родительским процессом, а запущенный – дочерним. Так вот, когда дочерний процесс выполнит свою задачу, он умирает, держа в руках табличку со статусом завершения, мол, всё ок или не ок. То есть передаёт родительскому процессу статус выхода. Такой процесс называется зомби. При этом родительский процесс должен прочитать эту табличку, тем самым отпустить зомби процесс с миром. Зомби процессы не используют никаких ресурсов, просто пока родитель не прочтёт

статус, они всё ещё числятся с таблице процессов. Но если родительская программа плохо написана, постоянно создаёт дочерние процессы, с которыми не прощается, то со временем таблица будет заполняться зомби процессами. А количество процессов в таблице ограничено. Для 32-битных систем максимально может быть примерно 32 тысячи записей, а для 64 битных – 4 миллиона. Но это значение можно поставить и поменьше. Увидеть текущий максимум можно взглянув на файл `cat /proc/sys/kernel/pid_max`. Обычно, для пользователей ставят своё ограничение на количество процессов – это можно увидеть с помощью команды `ulimit -u`. Ограничение у пользователей ставится, чтобы какой-то один пользователь не забил всю таблицу, либо не запустил слишком много процессов. Ведь когда таблица забита, например, теми же зомби процессами – то ничего сделать с системой не получится, потому что не получится создать новый процесс. Остаётся разве что по жёсткому перезагрузить систему.



The image shows two terminal windows. The top window is editing `/etc/security/limits.conf` with GNU nano 2.9.8. The content is as follows:

Group	Type	Resource	Limit
##*	soft	core	0
##*	hard	rss	10000
#@student	hard	nproc	20
#@faculty	soft	nproc	20
#@faculty	hard	nproc	50
#ftp	hard	nproc	0
#@student	-	maxlogins	4

The bottom window is editing `/home/user/.bash_profile` with GNU nano 2.9.8. The content is:

```
14 # Limits
15 ulimit -u 1024
```

Чтобы ограничить количество процессов у пользователя, нужно в домашней директории этого пользователя в файле `~/.bash_profile` выставить значение, например, `ulimit -u 1024`. Но обычно этот файл может редактировать пользователь, и, если у вас много пользователей или вы не доверяете ему, то, как правило, ограничения устанавливаются в файле `/etc/security/limits.conf`, либо внутри директории `/etc/security/limits.d/` создаётся файл, заканчивающийся на `.conf`. Пример есть в файле `/etc/security/limits.conf`. Нас интересует параметр `nproc` – number of processes – количество процессов. Это всё превентивные меры, которые защищают работу самой системы. Но если всё же у какого-то пользователя забились процессы и он не может запускать новые, то нужно избавляться от родительского процесса. Мы это рассмотрим чуть позже.

```
top - 18:59:13 up 1 min, 1 user, load average: 1.34, 0.43, 0.15
Tasks: 250 total, 3 running, 246 sleeping, 0 stopped, 1 zombie
%Cpu(s): 0.4 us, 0.4 sy, 0.0 ni, 98.6 id, 0.4 wa, 0.4 hi, 0.0 si, 0.0 st
MiB Mem : 1829.6 total, 170.5 free, 1144.1 used, 515.0 buff/cache
MiB Swap: 2048.0 total, 2032.0 free, 16.0 used. 512.1 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3336	root	29	9	187536	8264	3700	R	2.0	0.4	0:00.06	(coredump)
2588	user	20	0	2875352	301744	111764	S	1.3	16.1	0:03.80	gnome-she+
1	root	20	0	187532	13344	8804	S	0.7	0.7	0:00.95	systemd
973	root	16	-4	143088	2712	2100	S	0.3	0.1	0:00.01	auditd
1024	polkitd	20	0	1635812	25924	18528	S	0.3	1.4	0:02.24	polkitd
1032	dbus	20	0	72328	7060	4148	S	0.3	0.4	0:00.46	dbus-daem+
3206	user	20	0	736524	46976	36552	S	0.3	2.5	0:00.23	gnome-ter+
3304	user	20	0	273768	4888	4036	R	0.3	0.3	0:00.09	top

На третьей строчке немного информации про использование процессора. Она отображается в процентах. По сути это процент времени, которое процессор потратил на те или иные задачи в промежуток времени обновления информации в top, по умолчанию это 3 секунды. Немного запутано, но на первом примере, надеюсь, станет понятнее.

1. us – это user cpu time, т.е. время, потраченное на user space. Под user space подразумевается пользовательское пространство. И пусть вас не путает слово user, речь идёт о всех программах, кроме ядра. Для ядра есть пространство ядра – kernel space. Допустим, если взять веб сервер, то сама программа веб сервера, браузер, всякие утилиты – всё это user space. А ядро при этом отвечает за работу с сетью – это kernel space. Так вот, первый параметр – время, выделенное процессором на работу user space программ. Допустим, из 3 секунд полторы секунды процессор потратил на выполнение кода веб сервера – то тут значение будет 50%. Но нужно понимать, что процессор не уделяет все эти полторы секунды чисто одному процессу, какие-то доли секунд на один процесс, какие-то доли секунд на другой, и вот суммарно на user-space программы тратится сколько-то процентов времени.

2. sy – system cpu time – собственно, время, потраченное на kernel space.

3. ni – nice cpu time – время, потраченное на процессы с низким приоритетом. Процессов в системе много, каждый из них выполняет свою задачу – какая-то из них важная и нужна срочно, какая-то нет. Например, у меня рендерится видео и тут мне по работе присылают кучу документов в сжатом архиве. Мне нужно всё это разархивировать, поработать и заархивировать. Но оба этих процесса - рендеринг и архивация/сжатие - требуют много процессорного времени, поэтому мне придётся либо остановить рендеринг, либо долго ждать, пока разархивируется файл. Но, используя приоритеты, я могу сказать, что пусть рендеринг займёт чуть больше времени, мне срочно нужно выполнить новый процесс. Для выставления приоритетов используется утилита nice и числа от -20 до 19. nice в переводе – вежливый, поэтому чем больше число, тем “вежливее” программа. Т.е. программа с приоритетом 19 будет уступать процессорное время другим процессам. Таким образом, -20 это наивысший приоритет, а 19 это низший приоритет.

```
user@centos8 ~$ nice
0
[user@centos8 ~]$ nice -n -1 firefox
nice: cannot set niceness: Permission denied
[user@centos8 ~]$ nice -n 5 firefox

user@centos8 ~$ ps -l | head -1
F S  UID  PID  PPID  C PRI NI ADDR SZ WCHAN  TTY          TIME CMD
[user@centos8 ~]$ ps -el | grep firefox
4 S  1000 11724 4145 17  85   5 - 778157 x64_sy pts/1    00:00:02 firefox
[user@centos8 ~]$ renice -n 10 11724
11724 (process ID) old priority 5, new priority 10
[user@centos8 ~]$
```

Приоритет по умолчанию – 0. Это можно увидеть просто выполнив команду `nice`. Причём, более высокий приоритет, например, -1, -2, обычный пользователь не может задать, а вот приоритет пониже можно. Например, чтобы запустить `firefox` с приоритетом 5, я пишу `nice -n 5 firefox`. Чтобы посмотреть текущий приоритет, я использую `ps -el`. Допустим, `ps -el | grep firefox`. Обратите внимание, что 5 указано в столбике NI – это `niceness` – вежливость. А слева от него написано PRI – приоритет – и он отличается от того, что мы указывали в утилите `nice`. Это потому, что в `nice` мы указываем желаемый приоритет программы, но во время работы ядро распределяет приоритеты по своей шкале. Поэтому параметр `priority` показывает текущий приоритет в ядре, а `nice` – заданный приоритет. Если же программа уже запущена, её вежливость можно изменить с помощью `renice`. Причём, всегда можно повысить вежливость, тем самым уменьшив приоритет, если, конечно, процесс запущен от вашего пользователя. Но для повышения приоритета нужны права суперпользователя. Пример утилиты `renice` - `renice -n 10 pid`. Обратите внимание, что здесь я использую идентификатор процесса.

4. `id` – от слова `idle` - время, проведённое в ожидании. Как вы видите, большую часть времени процессор простаивает в ожидании программ.

5. `wa` – `input output wait` – время, потраченное процессором на ожидание чтения или записи на диск. Например, какая-то программа работает и в какой-то момент ей нужно записать данные на диск. Это занимает время, и, пока это происходит, процессор простаивает. На самом деле не совсем простаивает, он может и другими процессами заняться, но, в любом случае, для данного процесса он простаивает.

6. `hi` – `hardware interrupts` – аппаратные прерывания. Когда какое-то оборудование, допустим, жёсткий диск или сетевая карта, пытаются сообщить процессору о каком-то событии, оно посылает процессору сигнал, называемый аппаратным прерыванием. Собственно, этот параметр в `top` – время процессора, потраченное для аппаратных прерываний.

7. `si` – `software interrupts` – это уже программные прерывания.

8. `st` – `steal time` – это параметр, относящийся к виртуальным машинам – и говорит он о том, какое время реальный процессор был недоступен виртуальной машине, потому что был занят гипервизором или другими виртуальными машинами.

```
top - 18:59:13 up 1 min, 1 user, load average: 1.34, 0.43, 0.15
Tasks: 250 total, 3 running, 246 sleeping, 0 stopped, 1 zombie
%Cpu(s): 0.4 us, 0.4 sy, 0.0 ni, 98.6 id, 0.4 wa, 0.4 hi, 0.0 si, 0.0 st
MiB Mem : 1829.6 total, 170.5 free, 1144.1 used, 515.0 buff/cache
MiB Swap: 2048.0 total, 2032.0 free, 16.0 used. 512.1 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3336	root	29	9	187536	8264	3700	R	2.0	0.4	0:00.06	(coredump)
2588	user	20	0	2875352	301744	111764	S	1.3	16.1	0:03.80	gnome-she+
1	root	20	0	187532	13344	8804	S	0.7	0.7	0:00.95	systemd
973	root	16	-4	143088	2712	2100	S	0.3	0.1	0:00.01	auditd
1024	polkitd	20	0	1635812	25924	18528	S	0.3	1.4	0:02.24	polkitd
1032	dbus	20	0	72328	7060	4148	S	0.3	0.4	0:00.46	dbus-daem+
3206	user	20	0	736524	46976	36552	S	0.3	2.5	0:00.23	gnome-ter+
3304	user	20	0	273768	4888	4036	R	0.3	0.3	0:00.09	top

На четвёртой строчке у нас немного про оперативную память. Как видите, здесь написано MiB Mem – это мебибайты. Если вы раньше считали, что в одном килобайте 1024 байт, а в одном мегабайте – 1024 килобайт – это упрощение информатики. [На самом деле](#), в одном килобайте 1000 байт, а в одном мегабайте – 1000 килобайт. А вот 1024 относится к кибибайтам и мебибайтам. Но эти значения не сильно отличаются и можно догадаться, что 1800 с лишним мебибайт – это 2 гигабайта оперативки.

1. total – сколько всего оперативки
2. free – сколько оперативки свободно. Вас может смутить, что тут очень мало, но, не всё так просто и нужно смотреть третье значение
3. used – сколько оперативки используется. Казалось бы, математика простая – от общего числа отними используемое – и получится свободное – но тут не получается. Есть даже специальный [сайт](#), который говорит – не паникуйте, всё норм. Нужно понимать, что оперативка создана, чтобы её использовали, поэтому Linux старается выжать из неё максимум. Если она свободна, линукс использует её для кэша диска – то есть хранит часто запрашиваемые файлы с жёсткого диска в оперативке, тем самым всё работает быстрее. Если же какой-то программе понадобится оперативка, то она запросто отожмёт это места у кэша.
4. buff/cache – память, используемое для кэша.


```
user@centos8:~$ ulimit -v
unlimited
user@centos8:~$

GNU nano 2.9.8 /home/user/.bash_profile Modified

14 # Limits
15 ulimit -u 1024
16 ulimit -v 1000000

GNU nano 2.9.8 /etc/security/limits.conf

40 # - as - address space limit (KB)
41 # - maxlogins - max number of logins for this user
42 # - maxsyslogins - max number of logins on the system
```

Таким образом, если хотите знать, сколько оперативки использует ваш Linux – смотрите значение `used`. Вся остальная оперативка свободна для других программ. Кстати, как и с количеством процессов, вы можете ограничить количество виртуальной памяти, выделяемое каждому процессу. Для этого можете использовать `ulimit` с опцией `-v`, указав количество памяти в килобайтах, и записав в `~/.bash_profile`, чтобы всегда работало, либо в `/etc/security/limits.conf` прописать свое значение с параметром `as` – address space limit. Но виртуальная память состоит не только из оперативки, мы о ней ещё поговорим.

Ну и последняя строчка – `swap`, также называемый подкачкой. Это такой локальный филиал оперативки на жёстком диске. Когда оперативке перестаёт хватать места или ей требуется освободить пространство для более активных процессов, она может использовать специальное место на жёстком диске или `ssd`. Это может быть как специальный файл, так и целый раздел. Часто бывает полезно на некоторых серверах. Раньше это было полезно и на домашних компьютерах, когда не хватало оперативной памяти, но сейчас оперативки обычно много и всё это не так актуально. Подкачка ещё используется для спящего режима – но его размер должен быть больше оперативки, а это не всегда рационально, когда речь идёт про `ssd` и большое количество оперативки.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2730	root	20	0	413976	26520	5996	S	0.3	1.4	0:11.44	sssd_kcm
3206	user	20	0	740720	48104	32500	R	0.3	2.6	0:13.55	gnome-termi+
4052	user	20	0	273788	3748	2872	S	0.3	0.2	0:23.06	top
16346	user	20	0	273756	4944	4104	R	0.3	0.3	0:02.20	top
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0+
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	S	0.0	0.0	0:00.80	ksoftirqd/0
10	root	20	0	0	0	0	I	0.0	0.0	0:00.17	rcu_sched
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0

Ниже у нас таблица. Многое мы с вами уже разбирали:

PID - идентификатор процесса;

USER - пользователь, запустивший процесс;

PR - текущий приоритет в ядре. Кстати, обратите внимание, что у некоторых процессов вместо чисел написано rt – это real time – реальное время. Это означает, что такие процессы операционная система будет выполнять моментально, без всяких ожиданий выполнения других процессов. Как правило, это процессы, работающие с оборудованием. Есть даже специальные операционные системы реального времени, которые используются в промышленности, телекоммуникациях и прочих местах, где есть требования к скорости работы операционной системы с железом.

NI - niceness - вежливость процесса;

VIRT – виртуальная память. Мы говорили, что при запуске для процесса создаётся иллюзия, что этот процесс единственный в оперативке. Так вот, это осуществляется благодаря тому, что для каждого процесса создаётся адресное пространство. Представьте себе это как программу excel – у каждого процесса есть свой лист с таблицей, где можно заполнять данные. Есть адреса, допустим A1, B5 и т.п. - и у разных листов, т.е. процессов, адреса могут совпадать, но данные в них разные. Эти адреса ядро операционной системы связывает где-то с физическими адресами в оперативке, где-то со свопом на диске, а где-то с большими файлами на диске, которые система не загружает в оперативку. И вот все эти адреса для процесса находятся в одном листе. Лист – просто аналогия, а на деле это называется адресным пространством.

RES – сколько реальной оперативной памяти использует процесс. Отображается в килобайтах, как и виртуальная память. Столбик MEM показывает это же значение в процентах от всей оперативной памяти.

SHR – shared – потенциально сколько виртуальной памяти может быть разделено с другими процессами. Чтобы разные процессы не дублировали одни и те же данные, к примеру, библиотеки, процессы могут делиться частью виртуальной памяти.

S – статус процесса. S – это спит, R – работает, I – специфично для потоков ядра, ожидающих каких-то событий. Тут ещё может быть Z – зомби и пара других статусов.

CPU – использование процессора в процентах. По умолчанию это процент не от всего процессора, а от одного ядра. Т.е. тут значения могут быть больше 100%, допустим, 150%.

TIME+ - сколько времени процессор работал с этим процессом. Тут минуты:секунды.сотые секунд.

На самом деле, это далеко не все столбцы, которые может показать top. В принципе можно перенастроить программу так, чтобы она показывала другие столбцы, но разбирать все это займёт много времени. У top достаточно хороший ман, а мы с вами и так достаточно ударились в теорию. Давайте уже к практике.

```

user@centos8:~
File Edit View Search Terminal Help
Z,B,E,e Global: 'Z' colors; 'B' bold; 'E'/'e' summary/task memory scale
l,t,m Toggle Summary: 'l' load avg; 't' task/cpu stats; 'm' memory info
0,1,2,3,I Toggle: '0' zeros; '1/2/3' cpus or numa node views; 'I' Irix mode
f,F,X Fields: 'f'/'F' add/remove/order/sort; 'X' increase fixed-width

L,&,<,> . Locate: 'L'/'&' find/again; Move sort column: '<'/'>' left/right
R,H,V,J . Toggle: 'R' Sort; 'H' Threads; 'V' Forest view; 'J' Num justify
c,i,S,j . Toggle: 'c' Cmd name/line; 'i' Idle; 'S' Time; 'j' Str justify
x,y . Toggle highlights: 'x' sort field; 'y' running tasks
z,b . Toggle: 'z' color/mono; 'b' bold/reverse (only if 'x' or 'y')
u,U,o,0 . Filter by: 'u'/'U' effective/any user; 'o'/'0' other criteria
n,#,^0 . Set: 'n'/'#' max tasks displayed; Show: Ctrl+'0' other filter(s)
C,... . Toggle scroll coordinates msg for: up,down,left,right,home,end

k,r Manipulate tasks: 'k' kill; 'r' renice
d or s Set update interval
W,Y Write configuration file 'W'; Inspect other output 'Y'
q Quit
( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows,
Type 'q' or <Esc> to continue

```

И так, у нас есть таблица, и больше всего здесь нас интересуют 3 столбца – процент использования процессора, процент использования оперативки и идентификатор процесса. И так, если я хочу отсортировать по использованию процессора, я нажимаю shift+p. Чтобы отсортировать по оперативке, я нажимаю shift+m. Ну и как всегда, можно просто нажать h и будет подсказка по горячим клавишам.

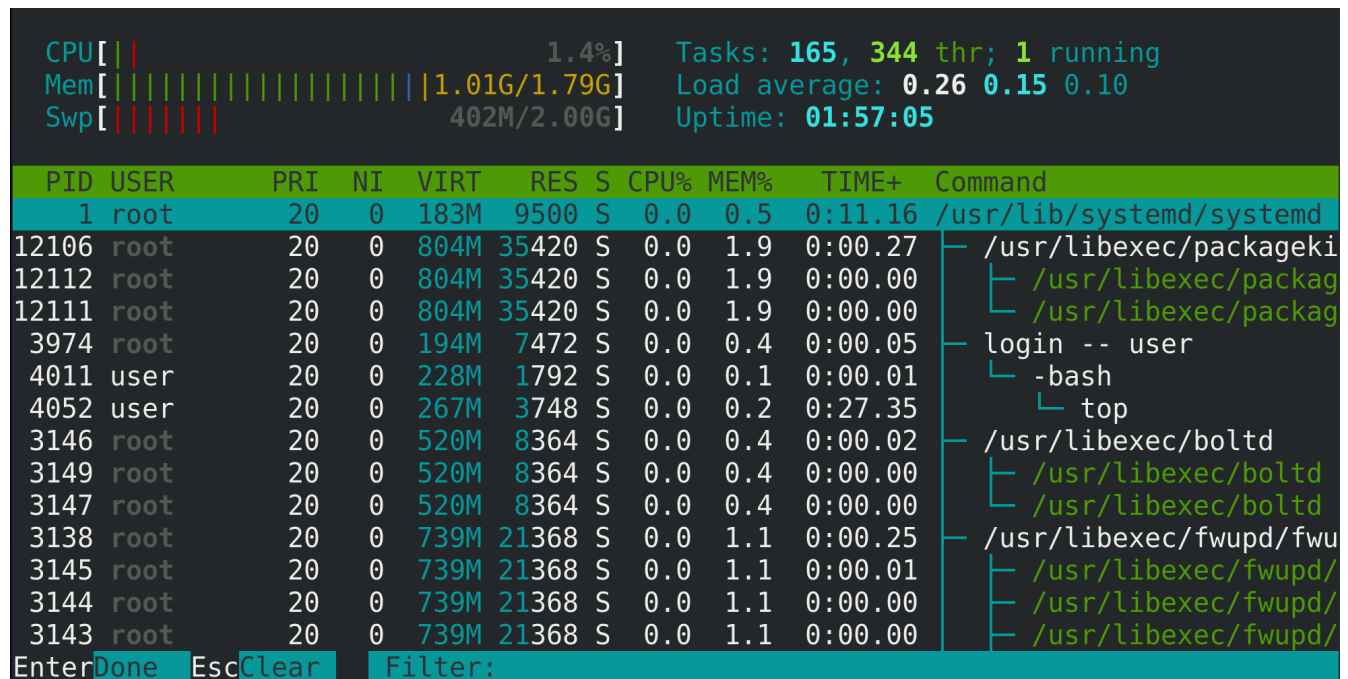
```

user@centos8:~
File Edit View Search Terminal Help
[user@centos8 ~]$ sudo dnf install httpd -y
Last metadata expiration check: 0:00:23 ago on Sun 29 Nov 2020 08:49:53 PM +04.
Dependencies resolved.
=====
Package                Architecture      Version           Repository        Size
=====
Installing:
httpd                   x86_64            2.2.0-6.el8       epel               112 k
Transaction Summary
=====

```

И хотя top даёт достаточно информации и предустановлен на многих UNIX-подобных системах, у него есть более простые интуитивно понятные аналоги. Как и в случае с vi, ситуации бывают разные и может случиться, что у вас не будет альтернатив, поэтому уметь хотя бы базово разбираться в top нужно. Но в повседневной жизни вы можете использовать, к

примеру, htop. Он у нас не предустановлен, поэтому давайте его установим – `sudo dnf install epel-release -y; sudo dnf install htop -y`.



The screenshot shows the htop interface. At the top, system statistics are displayed: CPU usage at 1.4%, Memory at 1.01G/1.79G, Swap at 402M/2.00G, Tasks at 165, 344 threads, 1 running, Load average at 0.26 0.15 0.10, and Uptime at 01:57:05. Below this is a table of running processes. The table has columns for PID, USER, PRI, NI, VIRT, RES, S, CPU%, MEM%, TIME+, and Command. The processes listed include systemd, packagekit, login, bash, top, boltd, fwupd, and various systemd services. At the bottom, there are buttons for 'Enter', 'Done', 'Esc', 'Clear', and a 'Filter:' input field.

PID	USER	PRI	NI	VIRT	RES	S	CPU%	MEM%	TIME+	Command
1	root	20	0	183M	9500	S	0.0	0.5	0:11.16	/usr/lib/systemd/systemd
12106	root	20	0	804M	35420	S	0.0	1.9	0:00.27	/usr/libexec/packageki
12112	root	20	0	804M	35420	S	0.0	1.9	0:00.00	/usr/libexec/packag
12111	root	20	0	804M	35420	S	0.0	1.9	0:00.00	/usr/libexec/packag
3974	root	20	0	194M	7472	S	0.0	0.4	0:00.05	login -- user
4011	user	20	0	228M	1792	S	0.0	0.1	0:00.01	-bash
4052	user	20	0	267M	3748	S	0.0	0.2	0:27.35	top
3146	root	20	0	520M	8364	S	0.0	0.4	0:00.02	/usr/libexec/boltd
3149	root	20	0	520M	8364	S	0.0	0.4	0:00.00	/usr/libexec/boltd
3147	root	20	0	520M	8364	S	0.0	0.4	0:00.00	/usr/libexec/boltd
3138	root	20	0	739M	21368	S	0.0	1.1	0:00.25	/usr/libexec/fwupd/fwu
3145	root	20	0	739M	21368	S	0.0	1.1	0:00.01	/usr/libexec/fwupd/
3144	root	20	0	739M	21368	S	0.0	1.1	0:00.00	/usr/libexec/fwupd/
3143	root	20	0	739M	21368	S	0.0	1.1	0:00.00	/usr/libexec/fwupd/

htop более интерактивен. Во первых, мы можем работать мышкой – для сортировки можно просто нажать на CPU или Mem. Снизу также есть кнопки с возможностями. Допустим, F2 – поменять внешний вид программы, поменять цвета, добавить столбцы – допустим уберем shared memory и сохраним. По F3 можем искать процессы по тексту. F4 – можем фильтровать. F5 покажет процессы в древовидной форме, чтобы увидеть родительские и дочерние процессы. Тот же niceness – обратите внимание, при нажатии F7 ничего не происходит, а при F8 вежливость увеличивается. Почему? F10 это выход, а вот F9 – kill. Если нажать – слева появится панель – послать сигнал и список сигналов. Но kill – это не часть htop, а вполне себе отдельная команда.

Попробуйте поиграться с лимитами – ограничьте у своего пользователя количество процессов или оперативной памяти у процессов, и посмотрите, что из этого выйдет. Не забудьте предварительно снять снэпшот виртуальной машины – как это делать я показывал в части с установкой CentOS. Выставив маленькое значение, возможно, вы не сможете даже зайти в систему. Постарайтесь исправить ситуацию без использования других пользователей или восстановления со снэпшота и расскажите в комментариях, как вы это сделали.