**4commands to check thread count per process (threads vs processes) in Linux**

**Table of Contents**

* [Threads vs Processes](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#Threads_vs_Processes)
* [Show threads per process](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#Show_threads_per_process)
  + [1. Using PID task](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#1_Using_PID_task)
  + [2. Using ps command](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#2_Using_ps_command)
  + [3. Using pstree command](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#3_Using_pstree_command)
  + [4. Using top command](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#4_Using_top_command)
* [Check thread count per process](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#Check_thread_count_per_process)
  + [1. Using PID status](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#1_Using_PID_status)
  + [2. Using ps command](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#2_Using_ps_command-2)
* [Check number of threads allowed in Linux system?](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#Check_number_of_threads_allowed_in_Linux_system)
* [What is the maximum processes count allowed in Linux?](https://www.golinuxcloud.com/check-threads-per-process-count-processes/#What_is_the_maximum_processes_count_allowed_in_Linux)

In Linux, some processes are divided into pieces called threads. In one liner, **threads are essentially just**[**processes**](https://www.golinuxcloud.com/check-if-script-is-already-running-shell-linux/)**with a shared address space on Linux**. In this article we will get some **brief overview on threads and processes**, also some examples to show threads per process, check thread count per process, check number of threads allowed, count threads and some more related topics.

**Threads vs Processes**

* A thread is very similar to a process, it has an identifier (TID, or thread ID), and the kernel schedules and runs threads just like processes.
* However, unlike separate processes, which usually do not share system resources such as memory and I/O connections with other processes, all threads inside a single process share their system resources and some memory.
* A process with one thread is single-threaded, and a process with more than one thread is multithreaded.
* All processes start out single-threaded. This starting thread is usually called the main thread. The main thread may then start new threads in order for the process to become multithreaded, similar to the way a process can call fork() to start a new process.
* The primary advantage of a multithreaded process is that when the process has a lot to do, threads can run simultaneously on multiple processors, potentially speeding up computation.
* Although you can also achieve simultaneous computation with multiple processes, threads start faster than processes, and it is often easier and/or more efficient for threads to intercommunicate using their shared memory than it is for processes to communicate over a channel such as a network connection or a pipe.

## Show threads per process

There are multiple commands and methods which you can use to show threads per process and count threads in Linux.

### 1. Using PID task

You can count threads with the list of available sub directories inside /proc/<PID>/task/. The count of total available sub-directories inside this part is directly proportional to the thread count per process for the provided PID.

For example to [check java](https://www.golinuxcloud.com/set-java-timezone-tzdata-openjdk-tzupdater/) thread count, I have a Java process for which you can see I have multiple sub-directories so it means this is a multi threaded process. using ls command under this path you can show threads per process for java

# ls /proc/$(pidof java)/task/

31161 31170 31175 31180 31185 31266 31274 31285 31290 31295 31301 31307

31165 31171 31176 31181 31186 31267 31276 31286 31291 31296 31302 31308

31167 31172 31177 31182 31241 31268 31279 31287 31292 31298 31303 38715

31168 31173 31178 31183 31260 31270 31283 31288 31293 31299 31304 42883

31169 31174 31179 31184 31265 31272 31284 31289 31294 31300 31306 47335

But then again I have another process for which as you can see I have single sub-directory hence we know this is a single thread process

# ls /proc/$(pgrep amsHelper)/task/

6164

### 2. Using ps command

You can also use “ps” command to show threads per process. With “ps” we can list LWP (Light Weight process) which depicts Thread ID of the respective process and **NWLP** (Number of Threads).

To show threads per process using ps command you can use below argument

-L Show threads, possibly with LWP and NLWP columns.

-e Select all processes

-f Do full-format listing

For example:

# ps -eLf | less

UID PID PPID LWP C NLWP STIME TTY TIME CMD

root 1 0 1 0 1 Nov08 ? 00:01:48 /usr/lib/systemd/systemd --switched-root --system --deserialize 22

root 2 0 2 0 1 Nov08 ? 00:00:00 [kthreadd]

root 4 2 4 0 1 Nov08 ? 00:00:00 [kworker/0:0H]

root 6 2 6 0 1 Nov08 ? 00:00:04 [ksoftirqd/0]

root 7 2 7 0 1 Nov08 ? 00:00:00 [migration/0]

<<Output trimmed>>

### 3. Using pstree command

You can also use pstree to show threads per process. Here as you see java thread count and check number of threads for java process

# pstree -pau -l -G -s 31161

systemd,1 --switched-root --system --deserialize 22

mqdtomcat-wdg,31160,watchdog /usr/bin/dtomcat-wdg start

mqjava,31161 -Xms300m -Xmx300m -XX:-UseLargePages -classpath /usr/share/tomcat/bin/bootstrap.jar:/usr/share/tomcat/bin/tomcat-juli.jar:/usr/share/java/commons-daemon.jar:/opt/watchdog/tomcat/lib/onends-tomcat.jar -Dcatalina.base=/opt/watchdog/tomcat -Dcatalina.home=/usr/share/tomcat -Djava.endorsed.dirs= -Djava.io.tmpdir=/opt/watchdog/tomcat/temp -Djava.util.logging.config.file=/opt/watchdog/tomcat/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager org.apache.catalina.startup.Bootstrap start

tq{java},31165

tq{java},31167

tq{java},31168

tq{java},31169

tq{java},31170

tq{java},31171

tq{java},31172

<<Output trimmed>>

### 4. Using top command

In top, by default we will not be able to see thread count per process. But when running top, it is possible to change which fields to display and add this column to print thread count per process can be added manually.

* Press **f**
* This will show a list of fields that top can display. The fields that are displayed in bold are the ones that top will display.
* Use the down arrow to navigate to “**nTH**” (Number of Threads).
* Press to select “**nTH**“
* Press ‘**s**‘ to sort on number of threads.
* Press ‘**q**‘ to display the data of threads count.

Next you should see a new column at the end of top command with number of thread (nTH) column to show threads per process

top - 15:24:41 up 2 days, 19:31, 2 users, load average: 0.46, 0.52, 0.52

Tasks: 219 total, 1 running, 218 sleeping, 0 stopped, 0 zombie

%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

KiB Mem : 13199273+total, 12801261+free, 1678324 used, 2301800 buff/cache

KiB Swap: 4189180 total, 4189180 free, 0 used. 12937532+avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND nTH

9 root 20 0 0 0 0 S 0.3 0.0 5:17.22 rcu\_sched 1

6357 ssrun 20 0 5310412 156988 13344 S 0.3 0.1 3:59.69 jsvc 52

51240 root 20 0 168472 2448 1628 R 0.3 0.0 0:00.04 top 1

1 root 20 0 193420 6408 2652 S 0.0 0.0 1:48.68 systemd 1

2 root 20 0 0 0 0 S 0.0 0.0 0:00.22 kthreadd 1

4 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:0H 1

## Check thread count per process

Next you can use the above explained commands to also **check thread count per process**by customizing them a little bit.

### 1. Using PID status

To check thread count per process you can use below command. For example here java thread count is 59 threads in my Linux environment

# cat /proc/$(pgrep java)/status | grep -i Threads

Threads: 59

While amsHelper process has single thread

# cat /proc/$(pgrep amsHelper)/status | grep -i Threads

Threads: 1

### 2. Using ps command

We used ps command to show threads per process and count threads, we can also use “ps” command to get **LWP** and **NLWP** details, which when combined with “wc” we can count threads per process.

To check thread count per process for a particular PID for example to check java thread count:

# ps -eL -q 31161 | wc -l

61

## Check number of threads allowed in Linux system?

Linux doesn’t have a separate threads per process limit, [just a limit on the total number of processes on the system](https://access.redhat.com/solutions/1420363). This value controls the maximum number of threads that can be created using fork(). During initialization the kernel sets this value such that even if the maximum number of threads is created

To check number of threads which Linux system can allow

# cat /proc/sys/kernel/threads-max

35000

The minimum number of threads that can be written to threads-max is 20.  
The maximum value that can be written to threads-max is given by the constant FUTEX\_TID\_MASK (0x3fffffff).  
If a value outside of this range is written to threads-max an error EINVAL occurs.

The default value depends on memory size. You can use threads-max to check number of threads allowed in Linux. You can increase thread count per process limit like this:

# echo 100000 > /proc/sys/kernel/threads-max

**IMPORTANT NOTE:**

The value written is checked against the available RAM pages. If the thread structures would occupy too much (more than 1/8th) of the available RAM pages threads-max is reduced accordingly.

There is also a limit on the number of processes (an hence threads) that a single user may create, see ulimit for details regarding these limits:

# ulimit -a | grep -i processes

max user processes (-u) 10000

Here, the system is able to create 35,000 threads/processes in total and a single user can create 10000 number of processes.

The logic is very simple here every CPU can execute 1 process at a time, if there are 8 cores that means 8 to 10 processes at a time can be executed easily without any stress but if number of running or runnable threads per CPU increases drastically then there will be performance issue.

## What is the maximum processes count allowed in Linux?

From the [proc man page](http://man7.org/linux/man-pages/man5/proc.5.html)

/proc/sys/kernel/pid\_max (since Linux 2.5.34)

This file specifies the value at which PIDs wrap around (i.e.,

the value in this file is one greater than the maximum PID).

PIDs greater than this value are not allocated; thus, the

value in this file also acts as a system-wide limit on the

total number of processes and threads. The default value for

this file, 32768, results in the same range of PIDs as on ear‐

lier kernels.

Verify the value for kernel.pid\_max

[root@server1 ~]# sysctl -a | grep kernel.pid\_max

kernel.pid\_max = 35000

Here I can execute 35,000 processes simultaneously in my system that can run in separate memory spaces.

To change the value of kernel.pid\_max to 65534:

# echo kernel.pid\_max = 65534 >> /etc/sysctl.conf

# sysctl -p

 Lastly I hope the steps from the article to show threads per process, check thread count per process, check number of threads allowed on Linux was helpful. So, let me know your suggestions and feedback using the comment section.