# Maximum number of threads per process in Linux?

What is the maximum number of threads that can be created by a process under Linux?

How (if possible) can this value be modified?

linux multithreading

asked Dec 5 '08 at 15:38
user60008667

## 12 Answers

Linux doesn't have a separate threads per process limit, just a limit on the total number of processes on the system (threads are essentially just processes with a shared address space on Linux) which you can view like this:

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

The default is the number of memory pages/4. You can increase this like:

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

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

answered Dec 5 '08 at 15:56
Robert Gamble
64.6k ● 16 ● 122 ● 128

3 The limit in /proc/sys/vm/max_map_count may limit the number of threads, too. It should be safe to increase that limit a lot if you hit it. – Mikko Rantalainen Jan 13 '12 at 11:50

1 Robert: Linux does implement per process limit indirectly. Check my answer for details ;) – codersofthedark Feb 9 '12 at 13:43

I am trying to change this on my ubuntu 12.04 and it not changing with your command. I also tried vi to change it, but I get `E667: Fsync failed` when I try to save on vi. – Siddharth May 4 '13 at 5:04

3 @dragosrsupercool the maximum thread are calculated using the total ram, no the virtual memory – c4f4t0r Nov 11 '13 at 19:39

The amount of stack size per thread (the default on your system) is more likely to be the limit than anything else. Reducing the per-thread stack size is a way to increase the total number of threads (although that's rarely a good idea). – Randy Howard Jan 23 at 18:22

This is WRONG to say that LINUX doesn't have a separate threads per process limit.

> **Linux implements max number of threads per process indirectly!!**

```
number of threads = total virtual memory / (stack size*1024*1024)
```

Thus, the number of threads per process can be increased by increasing total virtual memory or by decreasing stack size. But, decreasing stack size too much can lead to code failure due to stack overflow while max virtual memory is equals to the swap memory.

**Check you machine:**

Total Virtual Memory: `ulimit -v` (default is unlimited, thus you need to increase swap memory to increase this)

Total Stack Size: `ulimit -s` (default is 8Mb)

**Command to increase these values:**

```
ulimit -s newvalue

ulimit -v newvalue
```

*Replace new value with the value you want to put as limit.

**References:**

http://dustycodes.wordpress.com/2012/02/09/increasing-number-of-threads-per-process/

answered Feb 9 '12 at 13:39
codersofthedark
2,638 ●4 ●21 ●50

4 Except 3 little details: 1. Linux doesn't do this, the presence of stacks and the fact that memory and address space are of finite size has nothing to do with it. 2. You have to specify a thread's stack when creating it, this is irrespective of `ulimit -s`. It is very well possible (not sensible, but possible) to create as many threads as there are possible thread IDs. Under 64 bit Linux, it is even easily "possible" to create more threads than there are thread IDs (of course it's not possible, but as far as stack goes, it is). 3. Stack reserve, commit and VM are different things, esp with OC. – Damon Jun 21 '13 at 18:42

---

In practical terms, the limit is usually determined by stack space. If each thread gets a 1MB stack (I can't remember if that is the default on Linux), then you a 32-bit system will run out of address space after 3000 threads (assuming that the last gb is reserved to the kernel).

However, you'll most likely experience terrible performance if you use more than a few dozen threads. Sooner or later, you get too much context-switching overhead, too much overhead in the scheduler, and so on. (Creating a large number of threads does little more than eat a lot of memory. But a lot of threads with actual *work* to do is going to slow you down as they're fighting for the available CPU time)

What are you doing where this limit is even relevant?

edited Aug 27 '11 at 8:39
answered Dec 5 '08 at 16:03
jalf
182k ●36 ●257 ●492

3 1MB per thread for stack is pretty high, many programs don't need anywhere near this much stack space. The performance is going to be based on the number of *runnable* processes, not the number of threads that exist. I have a machine running right now with 1200+ threads with a load of 0.40. – Robert Gamble Dec 5 '08 at 16:21

8 the performance depends on what the the threads are doing. you can go much higher than a few dozen if they don't do much and hence less context-switching. – Corey Goldberg Mar 11 '09 at 14:09

2 default stacksize of thread in linux x86 is 2 Mb – osgx Feb 4 '10 at 0:11

stack is growing dynamically, only initial page is allocated off-the-bat – Michael Pankov Nov 4 '15 at 22:26

---

To retrieve it:

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

To set it:

```
echo 123456789 > /proc/sys/kernel/threads-max
```

123456789 = # of threads

edited Dec 5 '08 at 16:30
answered Dec 5 '08 at 15:50
Vincent Van Den Berghe
3,319 ●1 ●21 ●36

What's in /proc depends on the kernel, not the distro. – Paul Tomblin Dec 5 '08 at 15:55

Thanks, I removed the distro stuff. So I'm guessing this works on all 2.x kernels? – Vincent Van Den Berghe Dec 5 '08 at 16:31

---

@dragosrsupercool

Linux doesn't use the virtual memory to calculate the maximum of thread, but the physical ram installed on the system

```
 max_threads = totalram_pages / (8 * 8192 / 4096);
```

http://kavassalis.com/2011/03/linux-and-the-maximum-number-of-processes-threads/

kernel/fork.c

```
/* The default maximum number of threads is set to a safe
 * value: the thread structures can take up at most half
 * of memory.
 */
max_threads = mempages / (8 * THREAD_SIZE / PAGE_SIZE);
```

So thread max is different between every system, because the ram installed can be from different sizes, I know Linux doesn't need to increase the virtual memory, because on 32 bit we got 3 GB for user space and 1 GB for the kernel, on 64 bit we got 128 TB of virtual memory, that happen on Solaris, if you want increase the virtual memory you need to add swap space.

edited Aug 28 '15 at 10:52
Delimitry
2,275 ● 3 ● 13 ● 31

answered Nov 11 '13 at 19:16
c4f4t0r
657 ● 5 ● 18

---

proper 100k threads on linux:

```
ulimit -s  256
ulimit -i  120000
echo 120000 > /proc/sys/kernel/threads-max
echo 600000 > /proc/sys/vm/max_map_count
echo 200000 > /proc/sys/kernel/pid_max

 ./100k-pthread-create-app
```

answered Oct 4 '14 at 7:50
Vladimir Kunschikov
935 ● 1 ● 7 ● 11

2 Thank you, it finally allowed me to break through the 32k Java thread count. – berezovskiy Mar 11 at 23:12

---

Thread count limit: $ cat /proc/sys/kernel/threads-max

How it is calculated: max_threads = mempages / (8 * THREAD_SIZE / PAGE_SIZE); and: x86_64 page size (PAGE_SIZE) is 4K; Like all other architectures, x86_64 has a kernel stack for every active thread. These thread stacks are THREAD_SIZE (2*PAGE_SIZE) big;

for mempages : cat /proc/zoneinfo | grep spanned | awk '{totalpages=totalpages+$2} END {print totalpages}';

so actually the number is not related with limitation of thread memory stack size (ulimit -s). ps: thread memory stack limitation is 10M in my rhel VM, and for 1.5G memory, this VM can only afford 150 threads?

answered Feb 21 '14 at 6:19
Albert Kong
61 ● 1 ● 1

---

It probably shouldn't matter. You are going to get much better performance designing your algorithm to use a fixed number of threads (eg, 4 or 8 if you have 4 or 8 processors). You can do this with work queues, asynchronous IO, or something like libevent.

answered Dec 5 '08 at 16:18
twk
8,118 ● 13 ● 56 ● 87

---

Use `nbio` non-blocking i/o library or whatever, if you need more threads for doing I/O calls that block

edited Aug 27 '11 at 8:27
JVerstry
21.1k ● 49 ● 168 ● 306

answered Nov 16 '09 at 18:27
wefeqfw
39 ● 1

---

Depends on your system, just write a sample program [ by creating processes in a loop ] and check using ps axo pid,ppid,rss,vsz,nlwp,cmd. When it can no more create threads check nlwp count [ nlwp is the number threads ] voila you got your fool proof answer instead of going thru books

answered Jun 21 '13 at 18:17
resultsway
1,765 ● 1 ● 16 ● 29

---

We can see the maximum number of threads defined in the following file in linux cat /proc/sys/kernel/threads-max

(OR)

sysctl -a | grep threads-max

answered Jul 25 at 7:40
KrishnaKumar Madagani
1 ● 1

---

To set permanently,

```
vim /etc/sysctl.conf
```

**and add**

```
kernel.threads-max = "value"
```

answered Oct 17 at 15:07
Matteo Zocca
1 ● 1

---