max thread per process in linux

I wrote a simple program to calculate the maximum number of threads that a process can have in linux (Centos 5). here is the code:

```
int main()
{
    pthread_t thrd[400];
    for(int i=0;i<400;i++)
    {
        int err=pthread_create(&thrd[i],NULL,thread,(void*)i);
        if(err!=0)
            cout << "thread creation failed: " << i <<" error code: " << err <<
endl;
    }
    return 0;
}

void * thread(void* i)
{
    sleep(100);//make the thread still alive
    return 0;
}
```

I figured out that max number for threads is only 300!? What if i need more than that? I have to mention that pthread_create returns 12 as error code.

Thanks before

linux pthreads

asked Apr 12 '11 at 12:43
Hosi
170 ●2 ●2 ●16

6 If you need more than 300 threads you really should rethink your design – Erik Apr 12 '11 at 12:48

1 You shouldn't hit that limit whatever it is. You should create a pool of threads (possibly with a user-configured size). – khachik Apr 12 '11 at 13:10

@Erik & khachik: Now I'm just wondering how to do that if is necessary! but thanks about pool idea. – Hosi Apr 12 '11 at 13:20

Stacksize of 1mb is still to big. errcode 12 = out of mem. strerror() will print the error code by the way. Try 16k stacksize. – johnnycrash Apr 15 '11 at 1:00

4 Answers

There is **a thread limit for linux** and it can be modified runtime by writing desired limit to `/proc/sys/kernel/threads-max`. The default value is computed from the available system memory. In addition to that limit, there's also another limit: `/proc/sys/vm/max_map_count` which limits the maximum mmapped segments and at least recent kernels will mmap memory per thread. It should be safe to increase that limit a lot if you hit it.

The **limit you're hitting is lack of virtual memory** in 32bit operating system. Install a 64 bit linux if your hardware supports it and you'll be fine. I can easily start 30000 threads with a stack size of 8MB. The system has a single Core 2 Duo + 8 GB of system memory (I'm using 5 GB for other stuff in the same time) and it's running 64 bit Ubuntu with kernel 2.6.32. Note that memory overcommit (/proc/sys/vm/overcommit_memory) must be allowed because otherwise system would need at least 240 GB of committable memory (sum of real memory and swap space).

If you need lots of threads and cannot use 64 bit system your only choice is to minimize the memory usage per thread to conserve virtual memory. Start with requesting as little stack as you can live with.

edited Jan 13 '12 at 12:02

answered Jan 13 '12 at 10:31
Mikko Rantalainen
4,267 ●2 ●32 ●53

Your system limits may not be allowing you to map the stacks of all the threads you require. Look at `/proc/sys/vm/max_map_count`, and see this answer. I'm not 100% sure this is your problem, because most people run into problems at much larger thread counts.

answered Apr 12 '11 at 13:26
Adrian Cox
4,086 ● 25 ● 51

1 "/proc/sys/vm/max_map_count" is 65536 and I can't change it! – Hosi Apr 12 '11 at 13:56

Oh well, that isn't the answer either. Another thing to try is to find your stack size with `pthread_attr_getstacksize` and check that your limits allow you to `malloc` that many stacks. – Adrian Cox Apr 12 '11 at 14:13

getstacksize returns 10MB, I decreased it to 1MB but still the same error! – Hosi Apr 14 '11 at 10:34

I had also encountered the same problem when my number of threads crosses some threshold. It was because of the user level limit (number of process a user can run at a time) set to 1024 in /etc/security/limits.conf .

so check your /etc/security/limits.conf and look for entry:-

username -/soft/hard -nproc 1024

change it to some larger values to something 100k(requires sudo privileges/root) and it should work for you.

To learn more about security policy ,see http://linux.die.net/man/5/limits.conf.

edited Sep 28 '13 at 16:37 answered Sep 28 '13 at 15:49
Ankit Singhal
436 ● 4 ● 8

You will run out of memory too unless u shrink the default thread stack size. Its 10MB on our version of linux.

EDIT: Error code 12 = out of memory, so I think the 1mb stack is still too big for you. Compiled for 32 bit, I can get a 100k stack to give me 30k threads. Beyond 30k threads I get Error code 11 which means no more threads allowed. A 1MB stack gives me about 4k threads before error code 12. 10MB gives me 427 threads. 100MB gives me 42 threads. 1 GB gives me 4... We have 64 bit OS with 64 GB ram. Is your OS 32 bit? When I compile for 64bit, I can use any stack size I want and get the limit of threads.

Also I noticed if i turn the profiling stuff (Tools|Profiling) on for netbeans and run from the ide...I only can get 400 threads. Weird. Netbeans also dies if you use up all the threads.

Here is a test app you can run:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <signal.h>

// this prevents the compiler from reordering code over this COMPILER_BARRIER
// this doesnt do anything
#define COMPILER_BARRIER() __asm__ __volatile__ ("" ::: "memory")

sigset_t    _fSigSet;
volatile int    _cActive = 0;
pthread_t   thrd[1000000];

void * thread(void *i)
{
int nSig, cActive;

    cActive = __sync_fetch_and_add(&_cActive, 1);
    COMPILER_BARRIER();  // make sure the active count is incremented before
sigwait

    // sigwait is a handy way to sleep a thread and wake it on command
    sigwait(&_fSigSet, &nSig); //make the thread still alive

    COMPILER_BARRIER();  // make sure the active count is decrimented after
sigwait
    cActive = __sync_fetch_and_add(&_cActive, -1);
    //printf("%d(%d) ", i, cActive);
    return 0;
}

int main(int argc, char** argv)
{
pthread_attr_t attr;
int cThreadRequest, cThreads, i, err, cActive, cbStack;

    cbStack = (argc > 1) ? atoi(argv[1]) : 0x100000;
    cThreadRequest = (argc > 2) ? atoi(argv[2]) : 30000;

    sigemptyset(&_fSigSet);
    sigaddset(&_fSigSet, SIGUSR1);
    sigaddset(&_fSigSet, SIGSEGV);

    printf("Start\n");
    pthread_attr_init(&attr);
    if ((err = pthread_attr_setstacksize(&attr, cbStack)) != 0)
        printf("pthread_attr_setstacksize failed: err: %d %s\n", err,
strerror(err));

    for (i = 0; i < cThreadRequest; i++)
    {
        if ((err = pthread_create(&thrd[i], &attr, thread, (void*)i)) != 0)
        {
            printf("pthread_create failed on thread %d, error code: %d %s\n",
                     i, err, strerror(err));
            break;
        }
    }
    cThreads = i;

    printf("\n");

    // wait for threads to all be created, although we might not wait for
    // all threads to make it through sigwait
    while (1)
    {
        cActive = _cActive;
        if (cActive == cThreads)
            break;
        printf("Waiting A %d/%d,", cActive, cThreads);
        sched_yield();
    }

    // wake em all up so they exit
    for (i = 0; i < cThreads; i++)
        pthread_kill(thrd[i], SIGUSR1);

    // wait for them all to exit, although we might be able to exit before
    // the last thread returns
    while (1)
    {
        cActive = _cActive;
        if (!cActive)
            break;
        printf("Waiting B %d/%d,", cActive, cThreads);
        sched_yield();
    }

    printf("\nDone. Threads requested: %d.  Threads created: %d.
StackSize=%lfmb\n",
     cThreadRequest, cThreads, (double)cbStack/0x100000);
    return 0;
}
```

edited Apr 15 '11 at 1:44

answered Apr 13 '11 at 1:12
johnnycrash
2,908 ●1 ●20 ●31

setstacksize to 1MB and 1KB but still the same error! – Hosi Apr 14 '11 at 10:36

1mb too big! Try 16k. I can get 400 threads with 1mb, and 30k threads with 16k We have 64 bit os and 64 gb ram. – johnnycrash Apr 15 '11 at 1:01

Thanks "johnnycrash" for you code, but I'm not familiar with "__sync_fetch_and_add" and can't compile this test app. Would you mind give more information about it? Thanks – Hosi Apr 15 '11 at 10:48

and my OS is 32bit. – Hosi Apr 15 '11 at 11:10

1 __synch comes from gcc atomic operations. gcc.gnu.org/onlinedocs/gcc-4.1.2/gcc/Atomic-Builtins.html I used em to atomically increment/decriment a value without having to use mutexes (mutex = slow). We are using gcc on intel, when i compile 32 bit I have to use -m32 -march=i686. -march is needed when using the atomic builtins. When I compile to 64bit I don't have to use -m32 of course, but I also don't need -march=i686 which seems odd. We use 64 bit RedHat here, and we have either 32 or 64 GB ram. Are you getting error code 11 or 12? 11=no more threads allowed, 12 = out of mem. – johnnycrash Apr 15 '11 at 13:00