Linux Stack Sizes

TUXERA

Work at Tuxera Inc. Helsinki, Finland

Linux Kernel Developer - File System … c assembly

Linux Kernel Developer - Flash Memory c linux

I'm looking for a good description of stacks within the linux kernel, but I'm finding it surprisingly difficult to find anything useful.

I know that stacks are limited to 4k for most systems, and 8k for others. I'm assuming that each kernel thread / bottom half has its own stack. I've also heard that if an interrupt goes off, it uses the current thread's stack, but I can't find any documentation on any of this. What I'm looking for is how the stacks are allocated, if there's any good debugging routines for them (I'm suspecting a stack overflow for a particular problem, and I'd like to know if its possible to compile the kernel to police stack sizes, etc).

linux linux-kernel linux-device-driver

edited Jun 7 '11 at 20:04
meagar♦
137k 21 204 240

asked Jun 7 '11 at 20:02
John Ulvr
246 1 4 8

what kernel version are you working with? so we have a better idea of the kernel debug configuration options available to you. – William Tate Jun 7 '11 at 20:08

I don't seem to get this. Why isn't a debugger sufficient for the task ? – cnicutar Jun 7 '11 at 20:11

"bottom-halves" probably share the same stack. Also, "bottom-halves" disappeared a long time ago, now there are softirqs left. – ninjalj Jun 7 '11 at 20:48

1 Use the `ulimit -s` command. The result is in KiB. – Miles Rout Aug 31 '13 at 5:31

Are you talking about the stack of user processes, or kthreads? Userland see also: unix.stackexchange.com/questions/145557/… – Ciro Santilli 烏坎事件2016六四事件 法轮功 May 28 at 15:27

3 Answers

The reason that documentation is scarce is that it's an area that's quite architecture-dependent. The code is really the best documentation - for example, the `THREAD_SIZE` macro defines the (architecture-dependent) per-thread kernel stack size.

The stacks are allocated in `alloc_thread_info_node()`, or the architecture-specific override for that function (the `struct thread_info` always lives at the bottom of the stack). The stack pointer in the `struct task_struct` is updated in `dup_task_struct()`, which is called as part of cloning a thread.

The kernel does check for kernel stack overflows, by placing a canary value `STACK_END_MAGIC` at the end of the stack (immediately after the `struct thread_info` in memory). In the page fault handler, if a fault in kernel space occurs this canary is checked - see for example the x86 fault handler which prints the message `Thread overran stack, or stack corrupted` after the Oops message if the stack canary has been clobbered.

Of course this won't trigger on *all* stack overruns, only the ones that clobber the stack canary. However, you should always be able to tell from the Oops output if you've suffered a stack overrun - that's the case if the stack pointer is below `&threadinfo`.

edited Jul 25 '15 at 3:27

answered Jun 8 '11 at 6:58
caf
156k 15 196 327

TUXERA

Work at Tuxera Inc. Helsinki, Finland

Linux Kernel Developer - File System … c assembly

Linux Kernel Developer - Flash Memory c linux

You can determine the process stack size with the `ulimit` command. I get 8192 KiB on my system:

```
$ ulimit -s
8192
```

edited Aug 6 '14 at 14:14

answered Aug 31 '13 at 5:31
Miles Rout
701 8 20

Low quality answer, because question was about kernel. Quote *"stacks within the linux kernel"*. – catpnosis Oct 15 '15 at 18:43

@GeoffreyR. You think that the stack is only 8kB in size? That would be rather useless. – Miles Rout Apr 12 at 21:55

@MilesRout My bad, I think it concerned the kernel stack size. – Geoffrey R. Apr 18 at 11:15

For processes, you can control the stack size of processes via `ulimit` command (`-s` option). For threads, the default stack size varies a lot, but you can control it via a call to `pthread_attr_setstacksize()` (assuming you are using pthreads).

As for the interrupt using the userland stack, I somewhat doubt it, as accessing userland memory is a kind of a hassle from the kernel, especially from an interrupt routine. But I don't know for sure.

answered Jun 7 '11 at 20:22

vhallac
8,250 2 16 29
