



阅码场™

www.yomocode.com

多线程编程(4)

阅码场™

www.yomocode.com

麦当劳喜欢您来，喜欢您再来



扫描关注
Linux阅码场



多线程编程(4)

多线程与栈

4.1 线程的栈

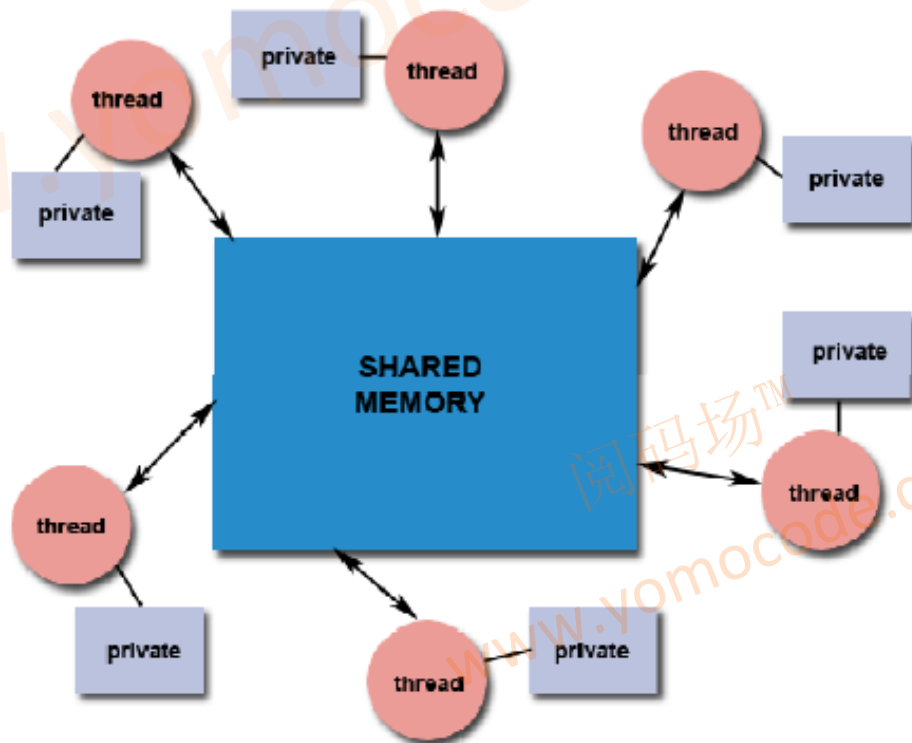
4.2 guard page

阅码场™

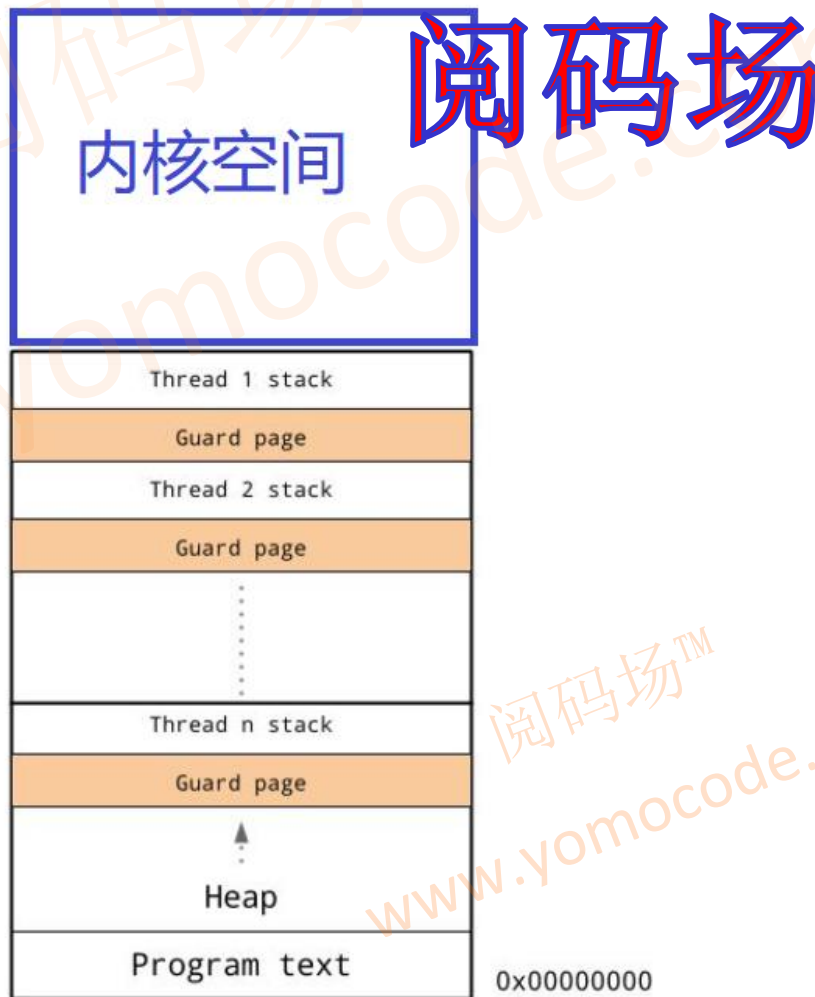
www.yomocode.com

线程栈的双重特性

栈：线程独有，保存其运行状态和局部自动变量的。栈在线程开始的时候初始化，每个线程的栈互相独立（但内存属于整个进程），因此，栈是 thread safe 的。



栈的guard page



POSIX库API

```
#include <pthread.h>
```

```
int pthread_attr_setguardsize(pthread_attr_t *attr, size_t guardsize);
```

```
int pthread_attr_getguardsize(const pthread_attr_t *attr, size_t  
*guardsize);
```

有意思的strace 的结果

```
mmap(NULL, 20480, PROT_NONE,  
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fbfcd14c000  
mprotect(0x7fbfcd14d000, 16384, PROT_READ|PROT_WRITE) = 0
```

```
mmap(NULL, 20480, PROT_NONE,  
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fbfcd147000  
mprotect(0x7fbfcd148000, 16384, PROT_READ|PROT_WRITE) = 0
```


pthread_create创建的线程 内存情况

`mprotect(R+W)`

`mmap(PROP_NONE)`



内核的一些改进

<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=1be7107fbe18eed3e319a6c3e83c78254b693acb>

← → ↺ git.kernel.org/pub/scm/linux/kernel/git/torvalds/... 📧 ☆

mm: **larger stack guard gap, between vmas**

Stack guard page is a useful feature to reduce a risk of stack smashing into a different mapping. We have been using a single page gap which is sufficient to prevent having stack adjacent to a different mapping. But this seems to be insufficient in the light of the stack usage in userspace. E.g. glibc uses as large as 64kB `alloca()` in many commonly used functions. Others use constructs like `gid_t buffer[NGROUPS_MAX]` which is 256kB or stack strings with `MAX_ARG_STRLEN`.

This will become especially dangerous for suid binaries and the default no limit for the stack size limit because those applications can be tricked to consume a large portion of the stack and a single glibc call could jump over the guard page. These attacks are not theoretical, unfortunately.

Make those attacks less probable by increasing the stack guard gap to 1MB (on systems with 4k pages; but make it depend on the page size because systems with larger base pages might cap stack allocations in the `PAGE_SIZE` units) which should cover larger `alloca()` and VLA stack allocations. It is obviously not a full fix because the problem is somehow inherent, but it should reduce attack space a lot.

One could argue that the gap size should be configurable from userspace, but that can be done later when somebody finds that the new 1MB is wrong for some special case applications. For now, add a kernel command line option (`stack_guard_gap`) to specify the stack gap size (in page units).

Implementation wise, first delete all the old code for stack guard page: because although we could get away with accounting one extra page in a stack vma, accounting a larger gap can break userspace - case in point, a program run with `"ulimit -S -v 20000"` failed when the 1MB gap was counted for `RLIMIT_AS`; similar problems could come with `RLIMIT_MLOCK` and strict non-overcommit mode.

Instead of keeping gap inside the stack vma, maintain the stack guard gap as a gap between vmas: using `vm_start_gap()` in place of `vm_start`.

谢谢!

阅码场™

www.yomocode.com

阅码场™

www.yomocode.com



阅码场出品