

Do manual pages matter?

Michael Kerrisk

linux.conf.au, Sydney, 17 January 2007

www.kernel.org/pub/linux/docs/manpages
mtk-manpages@gmx.net

Outline

Background

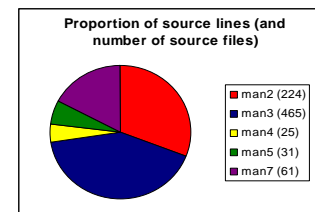
Man pages: a counter-argument
Man pages matter for kernel developers
Problems maintaining man pages
How to help

The *man-pages* project

- Project started 1993
- Documents Linux kernel-userland API...
- and (GNU) C library API
- Sections 2, 3, 4, 5, and 7 of manual pages
- Target audience: userland programmers...
- and kernel developers

Contents of *man-pages*

- As at *man-pages-2.44*:
 - ~800 man pages (== ~2000 printed pages)
 - 2: syscalls
 - 3: library functions (*glibc*)
 - 4: devices
 - 5: file formats
 - 7: overviews, etc



Background

Man pages: a counter-argument

Man pages matter for kernel developers

Problems maintaining man pages

How to help

“Documentation is fantasy: you have to read the source code to know the truth.”

Time!

- The kernel is **big**:
 - 2.6.19 kernel source (*. [chS]) is **7.3M lines**
- and constantly changing:
 - Typical Linux 2.6.x *diff-u* patch > **1M lines**

Reading the source doesn't cut it

- Reading the source gives the "right" answer
- but... too **slow** (and **hard**, especially for userland programmers)
- We just don't have the time...

We need summaries of the code

- Understanding of code must be mediated by *natural language* summaries
- Discussions
 - oral + email
 - Take place during development
 - but... not so useful later
- Documentation
 - most useful form of summary for later

Man pages do matter!

Background

Man pages: a counter-argument

Man pages matter for kernel developers

Problems maintaining man pages

How to help

Why man pages matter for kernel developers

- Publicity
- Identifying bugs
- Better testing (reducing # of released bugs)
- Better interface design
- Better interface consistency

Identifying bugs

- Software is an *implementation of an intention*
- *bug* == intention – implementation
- Without documentation, how do we know whether implementation matches intention?
- And how can we test?

Testing

- Problem: too many bugs in released interfaces
- Why? Insufficient testing before release

Documentation and Testing

- Documentation can help reduce bugs
- Evidence: the process can work in reverse...

Testing – example 1

inotify

- File change notification API
- Appeared in kernel 2.6.13
- 2.6.16-rc timeframe, I wrote *inotify(7)*
- Testing: `IN_ONESHOT` had *never* worked
- Bug reported; fixed for 2.6.16

Testing – example 2

splice()

- transfer data between file descriptors without going through user space
- Appeared in kernel 2.6.17
- Simple test programs easily caused hangs
- Bug reported; fixed for 2.6.18

Testing: conclusions

- Documentation goes hand in hand with testing
- Documentation broadens range of testers
- Testers can determine if *implementation* == *intention*
- Good, early documentation → more & earlier testing → fewer released bugs

Interface design

- It's hard to design a good programming interfaces
- Getting design wrong is painful...
 - Using interface is difficult, and bug-prone
 - Difficult/impossible to change design

When interface design goes wrong

- dnotify* (kernel 2.4; file change notification)
- Many problems in interface design
 - Problems led to replacement by *inotify*
 - But... is the problem the developer(s)?
 - Or the process?

Interface design: man pages help

- Writing a man page (or other doc) can help with interface design
- Writing documentation leads to self-review by implementer(s)
- Documentation broadens audience who can understand and critique design

Interface consistency

- The problem: some new interfaces are inconsistent with existing similar interfaces
- Man pages can be used as a reference when designing new interfaces

Interface consistency: right

- mbind(MPOL_MF_MOVE_ALL)***
- NUMA memory binding interface
 - Requires privilege (`CAP_xxx`)
 - Initial (-rc) implementation used `CAP_SYS_ADMIN`
 - Reading *capabilities(7)* showed that existing related APIs used `CAP_SYS_NICE`
 - Final implementation used `CAP_SYS_NICE`

Interface consistency: wrong (1)

- Various memory-related syscalls specify a *start* address + a *length*
- Some APIs (e.g., *mprotect(start, length, ...)*):
 - Require *start* to be multiple of page size
 - Round *length* up to next page boundary
- Some other APIs (e.g., *mlock(start, length)*):
 - Round *start* down to page size
 - Round *length* up to next page boundary
 - *mlock(4000, 6000)* affects bytes 0..12287

Interface consistency: wrong (2)

remap_file_pages(start, length, ...):

- Why settle just for inconsistent...
 - Round *start* down to page boundary
 - Round *length* **down** to page boundary(!)
- ... when you can also have bizarre:
 - What address range is affected by *remap_file_pages(4000, 6000, ...)*?

Background

Man pages: a counter-argument

Man pages matter for kernel developers

Problems maintaining man pages

How to help

Problems maintaining *man-pages*

- Much to do; too few people
- Many man pages yet to be written
- Many existing man pages are stale
- Kernel developers have much valuable knowledge, but are largely absent
- How to know if an interface has changed?
- How to know if a man page is broken?

Background

Man pages: a counter-argument

Man pages are useful for kernel developers

Problems maintaining man pages

How to help

How to help

- Just about anyone can help
- Kernel developers would benefit by helping
- How companies could help

Helping: anyone

- Read HOWTOHELP in *man-pages* tarball
 - List of missing pages
 - How to obtain list of FIXMEs
 - Tips on how to help in the most helpful way
- Latest tarball at:
<http://www.kernel.org/pub/linux/docs/manpages>

Helping: kernel developers

- Adding/changing an interface?...
- Write/update the manual page!
- Can't bear messing with *groff*?
 - Submit plain text!
- Please provide test programs...

Helping: kernel developers

- System call man pages belong in *man-pages*, not separate tarballs
- Many virtues in a consolidated set of man pages:
 - Formatting consistency
 - Single known address for man pages patches
 - Distributors know where to find manual page
 - Consistent interfaces...

Helping: kernel developers

"This [part of the] interface shouldn't be documented, because userland shouldn't be using it [it's only intended for use in libraries]."

- Library developers are in same position as everyone else
- "*no documentation*" doesn't always mean "*don't use this*"
- Best approach: document interface with warning about usage

A proposal for kernel developers

- Create and enforce a policy that **requires** interface changes to be accompanied by documentation and test programs

Before saying no...

- Consider that good documentation can help prevent:
 - Poorly designed/inconsistent interfaces
 - Bugs in new and changed interfaces
- Look at long list of FIXMEs and missing pages
- There are kernel coding standards; why not documentation (and testing) standards?

Helping: companies/organisations

- Fund a *man-pages* maintainer
 - Write/update pages
 - Vet patches
 - Test new interfaces
 - Track standards work (POSIX.1-200x/SUSv4 and beyond)
 - Write/choose a style guide
 - Maintain a website

`mtk-manpages@gmx.net`
Michael Kerrisk

Thanks!

`www.kernel.org/pub/linux/docs/manpages`