# Linux Kernel Tinification

Josh Triplett
josh@joshtriplett.org

Linux Plumbers Conference 2014

100%

ONE HUNDRED PERCENT

COMPREHENSIVE
AUTHORITATIVE
WHAT YOU NEED

ONE HUNDRED PERCENT

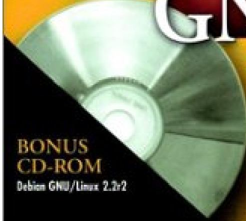
"Steve Hunger's book is the most comprehensive and up-to-date guide to Debian GNU/Linux in print."
—Branden Robinson, Debian Developer

**Master** Linux system administration

**Discover** the power of Debian's package management system

**Build** a network and set up Linux servers

# Debian® GNU/Linux® Bible



BONUS CD-ROM

Debian GNU/Linux 2.2r2

Steve Hunger

Foreword by Ian Murdock, Founder of Debian and now Cofounder of Progeny Linux Systems

## boot-floppies

## two floppies and an Internet connection

2.2.19 - 977k compressed

## debian-installer

one floppy and
an Internet connection

2.4.27 - 797k compressed

2.4.27 - 797k compressed
2.6.8 - 1073k compressed

“Linux runs on everything from cell phones to supercomputers”

## This is not an embedded system anymore

2GB RAM

16GB storage

## Original motivation

- Size-constrained bootloaders (why use GRUB?)
- x86 boot track: 32256 bytes

## Embedded systems

- Tiny flash part (1-8MB or smaller) for kernel and userspace
- CPU with onboard SRAM ($< 1024$kB)

## Compression

- vmlinuz is compressed
- Decompression stub for self-extraction

## Execute in place

- Don't load kernel into memory
- Run directly from flash
- Code and read-only data read from flash
- Read-write data in memory

## Execute in place

- Don’t load kernel into memory
- Run directly from flash
- Code and read-only data read from flash
- Read-write data in memory
- Minimizes memory usage

## Execute in place

- Don’t load kernel into memory
- Run directly from flash
- Code and read-only data read from flash
- Read-write data in memory
- Minimizes memory usage
- Precludes compression

## Configuring a minimal kernel

| **Configuration** | **Compressed** | **Uncompressed** |
|---|---:|---:|
| `make defconfig` | 5706k | 16532k |

## Configuring a minimal kernel

| **Configuration** | **Compressed** | **Uncompressed** |
|---|---:|---:|
| `make defconfig` | 5706k | 16532k |
| `make allnoconfig` | 503k | 1269k |

## Configuring a minimal kernel

| **Configuration** | **Compressed** | **Uncompressed** |
|---|---:|---:|
| `make defconfig` | 5706k | 16532k |
| `make allnoconfig` | 503k | 1269k |

- 3.15-rc1: `allnoconfig` automatically disables options behind `EXPERT` and `EMBEDDED`

## Configuring a minimal kernel

| Configuration | Compressed | Uncompressed |
|---|---|---|
| `make defconfig` | 5706k | 16532k |
| `make allnoconfig` | 503k | 1269k |

- 3.15-rc1: `allnoconfig` automatically disables options behind `EXPERT` and `EMBEDDED`
- 3.17-rc1: `tinyconfig`: enable `CC_OPTIMIZE_FOR_SIZE`, `OPTIMIZE_INLINING`, `KERNEL_XZ`, `SLOB`, `NOHIGHMEM`,

## Configuring a minimal kernel

| **Configuration** | **Compressed** | **Uncompressed** |
|---|---:|---:|
| `make defconfig` | 5706k | 16532k |
| `make allnoconfig` | 503k | 1269k |
| `make tinyconfig` | 346k | 1048k |

- 3.15-rc1: `allnoconfig` automatically disables options behind `EXPERT` and `EMBEDDED`
- 3.17-rc1: `tinyconfig`: enable `CC_OPTIMIZE_FOR_SIZE`, `OPTIMIZE_INLINING`, `KERNEL_XZ`, `SLOB`, `NOHIGHMEM`,

## Configuring a minimal kernel

| Configuration | Compressed | Uncompressed |
|---|---|---|
| `make defconfig` | 5706k | 16532k |
| `make allnoconfig` | 503k | 1269k |
| `make tinyconfig` | 346k | 1048k |

- 3.15-rc1: `allnoconfig` automatically disables options behind `EXPERT` and `EMBEDDED`
- 3.17-rc1: `tinyconfig`: enable `CC_OPTIMIZE_FOR_SIZE`, `OPTIMIZE_INLINING`, `KERNEL_XZ`, `SLOB`, `NOHIGHMEM`,
- Manually simulated "tinyconfig" on older kernels for size comparisons

## Configuring a minimal useful kernel

| **Configuration** | **Compressed** | **Uncompressed** |
|---|---:|---:|
| `make tinyconfig` | 346k | 1048k |

## Configuring a minimal useful kernel

| **Configuration** | **Compressed** | **Uncompressed** |
|---|---:|---:|
| `make tinyconfig` | 346k | 1048k |
| + ELF support | +2k | +4k |

## Configuring a minimal useful kernel

| Configuration | Compressed | Uncompressed |
|---|---:|---:|
| `make tinyconfig` | 346k | 1048k |
| + ELF support | +2k | +4k |
| + modules | +18k | +53k |

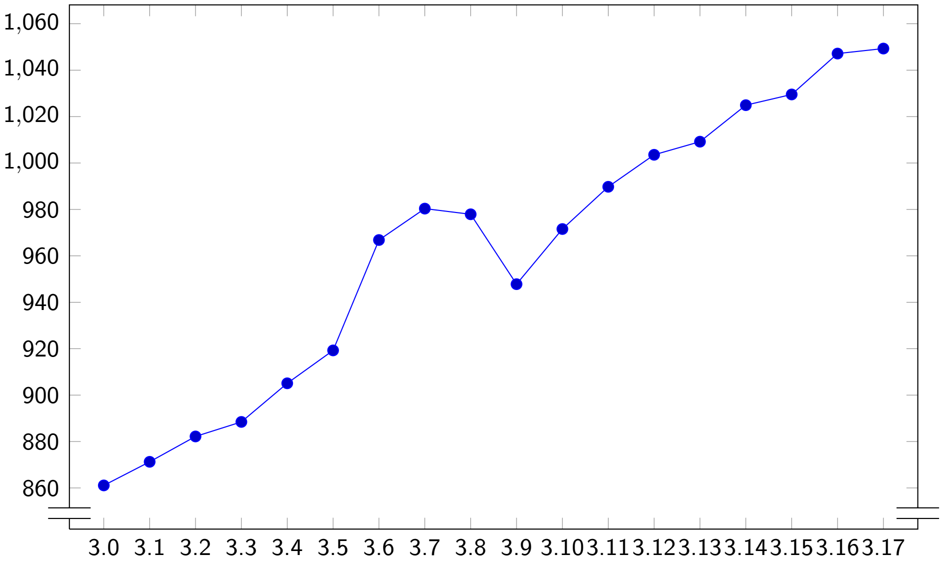
## Configuring a minimal useful kernel

| Configuration | Compressed | Uncompressed |
|---|---:|---:|
| `make tinyconfig` | 346k | 1048k |
| + ELF support | +2k | +4k |
| + modules | +18k | +53k |
| + initramfs | +32k | +37k |

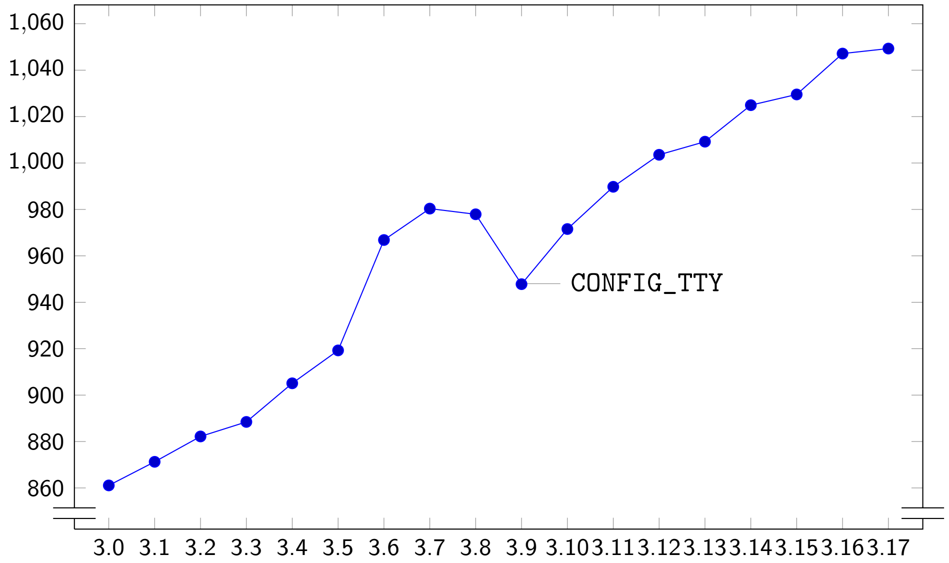
## Configuring a minimal useful kernel

| Configuration | Compressed | Uncompressed |
|---|---:|---:|
| `make tinyconfig` | 346k | 1048k |
| + ELF support | +2k | +4k |
| + modules | +18k | +53k |
| + initramfs | +32k | +37k |
| + flash storage | | |
| + filesystem | | |
| + networking | | |
| ... | | |

## minimum kernel size (kB) by kernel version



# minimum kernel size (kB) by kernel version



## Shrinking further

- Let's not give up and let "tiny" mean "proprietary RTOS"
- Linux could still go an order of magnitude smaller, at least

## Shrinking further

- Let's not give up and let "tiny" mean "proprietary RTOS"
- Linux could still go an order of magnitude smaller, at least
- Let's make the core as small as possible
- Leave maximum room for useful functionality

## nm --size-sort vmlinux

- Find large symbols for potential removal

```
00001000 d raw_data
00001000 d raw_data
00001210 r intel_tlb_table
00002000 D init_thread_union
00002000 r nhm_lbr_sel_map
00002000 r snb_lbr_sel_map
00002180 D init_tss
00003094 T real_mode_blob
00006000 b .brk.early_pgt_alloc
00100000 b .brk.pagetables
```

- 'r' is read-only, 'b' is bss, 'd' is data, 't' is text
- For memory usage, look at writable data and bss
- For compiled size, ignore bss

## nm --size-sort vmlinux

- Find large symbols for potential removal

```
00001000 d raw_data                    VDSO
00001000 d raw_data
00001210 r intel_tlb_table
00002000 D init_thread_union
00002000 r nhm_lbr_sel_map
00002000 r snb_lbr_sel_map
00002180 D init_tss
00003094 T real_mode_blob
00006000 b .brk.early_pgt_alloc
00100000 b .brk.pagetables
```

- 'r' is read-only, 'b' is bss, 'd' is data, 't' is text
- For memory usage, look at writable data and bss
- For compiled size, ignore bss

## nm --size-sort vmlinux

- Find large symbols for potential removal

```
00001000 d raw_data                    VDSO
00001000 d raw_data                    Another VDSO
00001210 r intel_tlb_table
00002000 D init_thread_union
00002000 r nhm_lbr_sel_map
00002000 r snb_lbr_sel_map
00002180 D init_tss
00003094 T real_mode_blob
00006000 b .brk.early_pgt_alloc
00100000 b .brk.pagetables
```

- 'r' is read-only, 'b' is bss, 'd' is data, 't' is text
- For memory usage, look at writable data and bss
- For compiled size, ignore bss

## nm --size-sort vmlinux

- Find large symbols for potential removal

```
00001000 d raw_data                  VDSO
00001000 d raw_data                  Another VDSO
00001210 r intel_tlb_table
00002000 D init_thread_union         initial thread and stack
00002000 r nhm_lbr_sel_map
00002000 r snb_lbr_sel_map
00002180 D init_tss
00003094 T real_mode_blob
00006000 b .brk.early_pgt_alloc
00100000 b .brk.pagetables
```

- 'r' is read-only, 'b' is bss, 'd' is data, 't' is text
- For memory usage, look at writable data and bss
- For compiled size, ignore bss

## nm --size-sort vmlinux

- Find large symbols for potential removal

```
00001000 d raw_data                  VDSO
00001000 d raw_data                  Another VDSO
00001210 r intel_tlb_table
00002000 D init_thread_union         initial thread and stack
00002000 r nhm_lbr_sel_map           tiny/disable-perf (-147k)
00002000 r snb_lbr_sel_map           tiny/disable-perf
00002180 D init_tss
00003094 T real_mode_blob
00006000 b .brk.early_pgt_alloc
00100000 b .brk.pagetables
```

- 'r' is read-only, 'b' is bss, 'd' is data, 't' is text
- For memory usage, look at writable data and bss
- For compiled size, ignore bss

## nm --size-sort vmlinux

- Find large symbols for potential removal

```
00001000 d raw_data                VDSO
00001000 d raw_data                Another VDSO
00001210 r intel_tlb_table
00002000 D init_thread_union       initial thread and stack
00002000 r nhm_lbr_sel_map         tiny/disable-perf (-147k)
00002000 r snb_lbr_sel_map         tiny/disable-perf
00002180 D init_tss                tiny/no-io (-9k)
00003094 T real_mode_blob
00006000 b .brk.early_pgt_alloc
00100000 b .brk.pagetables
```

- 'r' is read-only, 'b' is bss, 'd' is data, 't' is text
- For memory usage, look at writable data and bss
- For compiled size, ignore bss

## nm --size-sort vmlinux

- Find large symbols for potential removal

```
00001000 d raw_data                  VDSO
00001000 d raw_data                  Another VDSO
00001210 r intel_tlb_table
00002000 D init_thread_union         initial thread and stack
00002000 r nhm_lbr_sel_map           tiny/disable-perf (-147k)
00002000 r snb_lbr_sel_map           tiny/disable-perf
00002180 D init_tss                  tiny/no-io (-9k)
00003094 T real_mode_blob            copied to low mem
00006000 b .brk.early_pgt_alloc
00100000 b .brk.pagetables
```

- 'r' is read-only, 'b' is bss, 'd' is data, 't' is text
- For memory usage, look at writable data and bss
- For compiled size, ignore bss

## nm --size-sort vmlinux

- Find large symbols for potential removal

```
00001000 d raw_data                VDSO
00001000 d raw_data                Another VDSO
00001210 r intel_tlb_table
00002000 D init_thread_union       initial thread and stack
00002000 r nhm_lbr_sel_map         tiny/disable-perf (-147k)
00002000 r snb_lbr_sel_map         tiny/disable-perf
00002180 D init_tss                tiny/no-io (-9k)
00003094 T real_mode_blob          copied to low mem
00006000 b .brk.early_pgt_alloc    .bss
00100000 b .brk.pagetables         .bss
```

- 'r' is read-only, 'b' is bss, 'd' is data, 't' is text
- For memory usage, look at writable data and bss
- For compiled size, ignore bss

## nm --size-sort vmlinux

- Find large symbols for potential removal

```
00001000 d raw_data                VDSO
00001000 d raw_data                Another VDSO
00001210 r intel_tlb_table         Hmmmm...
00002000 D init_thread_union       initial thread and stack
00002000 r nhm_lbr_sel_map         tiny/disable-perf (-147k)
00002000 r snb_lbr_sel_map         tiny/disable-perf
00002180 D init_tss                tiny/no-io (-9k)
00003094 T real_mode_blob          copied to low mem
00006000 b .brk.early_pgt_alloc    .bss
00100000 b .brk.pagetables         .bss
```

- 'r' is read-only, 'b' is bss, 'd' is data, 't' is text
- For memory usage, look at writable data and bss
- For compiled size, ignore bss

## intel_tlb_table

- `git grep intel_tlb_table`

## intel_tlb_table

- git grep intel_tlb_table

```
static const struct _tlb_table intel_tlb_table[] = {
{ 0x01, TLB_INST_4K, 32, " TLB_INST 4 KByte pages ..." },
{ 0x02, TLB_INST_4M, 2,  " TLB_INST 4 MByte pages ..." },
/* ... 34 entries total ... */
```

## intel_tlb_table

- git grep intel_tlb_table

```
static const struct _tlb_table intel_tlb_table[] = {
{ 0x01, TLB_INST_4K, 32, " TLB_INST 4 KByte pages ..." },
{ 0x02, TLB_INST_4M, 2,  " TLB_INST 4 MByte pages ..." },
/* ... 34 entries total ... */

struct _tlb_table {
        unsigned char descriptor;
        char tlb_type;
        unsigned int entries;
        /* unsigned int ways; */
        char info[128];
};
```

## intel_tlb_table

- `git grep intel_tlb_table`

```
static const struct _tlb_table intel_tlb_table[] = {
{ 0x01, TLB_INST_4K, 32, " TLB_INST 4 KByte pages ..." },
{ 0x02, TLB_INST_4M, 2,  " TLB_INST 4 MByte pages ..." },
/* ... 34 entries total ... */

struct _tlb_table {
        unsigned char descriptor;
        char tlb_type;
        unsigned int entries;
        /* unsigned int ways; */
        char info[128];
};
```

- $34 * 128 = 4352$ bytes (0x1100)

## Shrinking `intel_tlb_table`

- Kconfig to remove human-readable descriptions?

## Shrinking `intel_tlb_table`

- Kconfig to remove human-readable descriptions?
- Absolutely nothing references those descriptions!

## Shrinking `intel_tlb_table`

- Kconfig to remove human-readable descriptions?
- Absolutely nothing references those descriptions!
- Just delete the info field
- Make the descriptions comments

## Shrinking `intel_tlb_table`

- Kconfig to remove human-readable descriptions?
- Absolutely nothing references those descriptions!
- Just delete the info field
- Make the descriptions comments
- How much did we save?

## `scripts/bloat-o-meter`

- Compare symbol sizes between two kernels
- Similar to diffstat
- `scripts/bloat-o-meter vmlinux-old vmlinux-new`

## scripts/bloat-o-meter

- Compare symbol sizes between two kernels
- Similar to diffstat
- `scripts/bloat-o-meter vmlinux-old vmlinux-new`

```
add/remove: 0/0 grow/shrink: 0/2 up/down: 0/-4361 (-4361)
function                old     new   delta
intel_detect_tlb        876     867      -9
intel_tlb_table        4624     272   -4352
```

## TLB round 2

```
struct _tlb_table {
        unsigned char descriptor;
        char tlb_type;
        unsigned int entries;
};
```

- All values for `entries` fit in a u16
- Result is copied into a u16 after lookup
- Wastes 4 bytes per entry (including padding)

## TLB round 2

```
struct _tlb_table {
        unsigned char descriptor;
        char tlb_type;
        unsigned int entries;
};
```

- All values for `entries` fit in a u16
- Result is copied into a u16 after lookup
- Wastes 4 bytes per entry (including padding)

```
add/remove: 0/0 grow/shrink: 0/2 up/down: 0/-146 (-146)
function                old     new   delta
intel_detect_tlb        867     857     -10
intel_tlb_table         272     136    -136
```

## TLB round 3

- We’ve just saved 4.5k in every kernel
- Can we do even better for embedded kernels?

## TLB round 3

- We've just saved 4.5k in every kernel
- Can we do even better for embedded kernels?
- Why do we decode the TLB, anyway?

## TLB round 3

- We've just saved 4.5k in every kernel
- Can we do even better for embedded kernels?
- Why do we decode the TLB, anyway?
- A single printk at boot time

## TLB round 3

- We've just saved 4.5k in every kernel
- Can we do even better for embedded kernels?
- Why do we decode the TLB, anyway?
- A single printk at boot time
- `#ifndef CONFIG_PRINTK`

## TLB round 3

- We've just saved 4.5k in every kernel
- Can we do even better for embedded kernels?
- Why do we decode the TLB, anyway?
- A single printk at boot time
- `#ifndef CONFIG_PRINTK`

```
add/remove: 0/3 grow/shrink: 0/0 up/down: 0/-1215 (-1215)
function                old     new   delta
intel_tlb_table         136       -    -136
cpu_detect_tlb_amd      222       -    -222
intel_detect_tlb        857       -    -857
```

## TLB summary

```
add/remove: 0/3 grow/shrink: 0/0 up/down: 0/-5722 (-5722)
function                old     new   delta
cpu_detect_tlb_amd  222       -    -222
intel_detect_tlb    876       -    -876
intel_tlb_table    4624       -   -4624
```

- 4.5k saved on every kernel
- 1.2k more saved on embedded kernels
- Patches in tinification tree, tiny/tlb branch

## syscalls

- Current Linux (on 32-bit x86) has ∼353 syscalls
- /bin/true uses ∼11 (less if static)
- Embedded systems fall somewhere in the middle

## syscalls

- Current Linux (on 32-bit x86) has ~353 syscalls
- /bin/true uses ~11 (less if static)
- Embedded systems fall somewhere in the middle
- `make tinyconfig` kernel has ~247
- Far too many unconditionally available syscalls

## A few unconditionally available syscalls

- adjtime/adjtimex and NTP support
- Older compatibility syscalls
- fallocate
- tee/splice
- kill and signal handling
- Scheduler configuration and priorities
- xattrs
- ptrace

## Removing syscalls

- Add Kconfig symbol for the syscall
  - `default y`
  - `bool "..." if EXPERT`

## Removing syscalls

- Add Kconfig symbol for the syscall
  - `default y`
  - `bool "..." if EXPERT`
- Add `cond_syscall(sys_foo);` to `kernel/sys_ni.c`

## Removing syscalls

- Add Kconfig symbol for the syscall
  - `default y`
  - `bool "..." if EXPERT`
- Add `cond_syscall(sys_foo);` to `kernel/sys_ni.c`
- Compile out the syscall entry point (SYSCALL_DEFINE)

## Removing syscalls

- Add Kconfig symbol for the syscall
  - `default y`
  - `bool "..." if EXPERT`
- Add `cond_syscall(sys_foo);` to `kernel/sys_ni.c`
- Compile out the syscall entry point (SYSCALL_DEFINE)
- Compile out the infrastructure

## Example: omitting madvise and fadvise

init/Kconfig:

```
+config ADVISE_SYSCALLS
+       bool "Enable madvise/fadvise syscalls" if EXPERT
+       default y
+       help
+         This option enables ...
```

## Example: omitting madvise and fadvise

init/Kconfig:

```
+config ADVISE_SYSCALLS
+       bool "Enable madvise/fadvise syscalls" if EXPERT
+       default y
+       help
+         This option enables ...
```

kernel/sys_ni.c:

```
+cond_syscall(sys_fadvise64);
+cond_syscall(sys_fadvise64_64);
+cond_syscall(sys_madvise);
```

## Example: Omitting madvise and fadvise (2)

mm/Makefile:

```
-obj-y := filemap.o mempool.o oom_kill.o fadvise.o \
+obj-y := filemap.o mempool.o oom_kill.o \
```

## Example: Omitting madvise and fadvise (2)

mm/Makefile:

```
-obj-y := filemap.o mempool.o oom_kill.o fadvise.o \
+obj-y := filemap.o mempool.o oom_kill.o \

+obj-$(CONFIG_ADVISE_SYSCALLS) += fadvise.o
```

## Example: Omitting madvise and fadvise (2)

mm/Makefile:

```
-obj-y := filemap.o mempool.o oom_kill.o fadvise.o \
+obj-y := filemap.o mempool.o oom_kill.o \

+obj-$(CONFIG_ADVISE_SYSCALLS) += fadvise.o

-mmu-$(CONFIG_MMU) := ... highmem.o madvise.o memory.o ...
+mmu-$(CONFIG_MMU) := ... highmem.o memory.o ...
```

## Example: Omitting madvise and fadvise (2)

mm/Makefile:

```
-obj-y := filemap.o mempool.o oom_kill.o fadvise.o \
+obj-y := filemap.o mempool.o oom_kill.o \

+obj-$(CONFIG_ADVISE_SYSCALLS) += fadvise.o

-mmu-$(CONFIG_MMU) := ... highmem.o madvise.o memory.o ...
+mmu-$(CONFIG_MMU) := ... highmem.o memory.o ...

+ifdef CONFIG_MMU
+        obj-$(CONFIG_ADVISE_SYSCALLS) += madvise.o
+endif
```

## Example: Omitting madvise and fadvise (2)

mm/Makefile:

```
-obj-y := filemap.o mempool.o oom_kill.o fadvise.o \
+obj-y := filemap.o mempool.o oom_kill.o \

+obj-$(CONFIG_ADVISE_SYSCALLS) += fadvise.o

-mmu-$(CONFIG_MMU) := ... highmem.o madvise.o memory.o ...
+mmu-$(CONFIG_MMU) := ... highmem.o memory.o ...

+ifdef CONFIG_MMU
+       obj-$(CONFIG_ADVISE_SYSCALLS) += madvise.o
+endif
```

- Saves 2.2k
- Merged during 3.18 merge window

## syscall infrastructure

- `uselib` (785 bytes)
  - In-kernel ELF library loader

## syscall infrastructure

- `uselib` (785 bytes)
  - In-kernel ELF library loader
- `iopl` and `ioperm` (9k)
  - Piles of task-switching code
  - Most of `init_tss` (seen in `nm --size-sort`)

## syscall infrastructure

- `uselib` (785 bytes)
  - In-kernel ELF library loader
- `iopl` and `ioperm` (9k)
  - Piles of task-switching code
  - Most of `init_tss` (seen in `nm --size-sort`)
- perf (147k)
  - Performance counter infrastructure
  - Complete x86 instruction decoder
  - Large per-CPU data tables
  - Hardware breakpoints

## Link-Time Optimization (LTO)

- Compile the entire kernel at once
- Cross-module optimization
- Automatically compile out unused code

## Link-Time Optimization (LTO)

- Compile the entire kernel at once
- Cross-module optimization
- Automatically compile out unused code
- Could reduce `#ifdef` logic to just top-level interfaces

## Compiler wishlist

- Transparently omitting struct fields
  - Compiler `__attribute__` on field declaration
  - Turn initialization and writes into no-ops
  - Error or dummy value on reads

## Compiler wishlist

- Transparently omitting struct fields
  - Compiler `__attribute__` on field declaration
  - Turn initialization and writes into no-ops
  - Error or dummy value on reads
  - Workaround: write all accesses as inline functions
  - Major code churn to switch from field to accessor functions

## Compiler wishlist

- Transparently omitting struct fields
  - Compiler `__attribute__` on field declaration
  - Turn initialization and writes into no-ops
  - Error or dummy value on reads
  - Workaround: write all accesses as inline functions
  - Major code churn to switch from field to accessor functions
- Constant folding through function pointer fields
  - Automatically notice no calls to a function pointer
  - Automatically omit it as above
  - Omit functions stored in that function pointer
  - Recurse

## Best practices

- Almost never add new unconditional code

## Best practices

- Almost never add new unconditional code
- Strings can be large!

## Best practices

- Almost never add new unconditional code
- Strings can be large!
- Decode-and-print infrastructure should be optional

## Best practices

- Almost never add new unconditional code
- Strings can be large!
- Decode-and-print infrastructure should be optional
- syscalls should be optional

## Best practices

- Almost never add new unconditional code
- Strings can be large!
- Decode-and-print infrastructure should be optional
- syscalls should be optional
- Infrastructure supporting those syscalls should be optional

## Best practices

- Almost never add new unconditional code
- Strings can be large!
- Decode-and-print infrastructure should be optional
- syscalls should be optional
- Infrastructure supporting those syscalls should be optional
- Improve toolchain to make tinification more automatic

## Best practices

- Almost never add new unconditional code
- Strings can be large!
- Decode-and-print infrastructure should be optional
- syscalls should be optional
- Infrastructure supporting those syscalls should be optional
- Improve toolchain to make tinification more automatic

Project list and tinification tree:

tiny.wiki.kernel.org