ELEC 424/553
Mobile & Embedded Systems

Lecture 8 - Introduction to Device Drivers

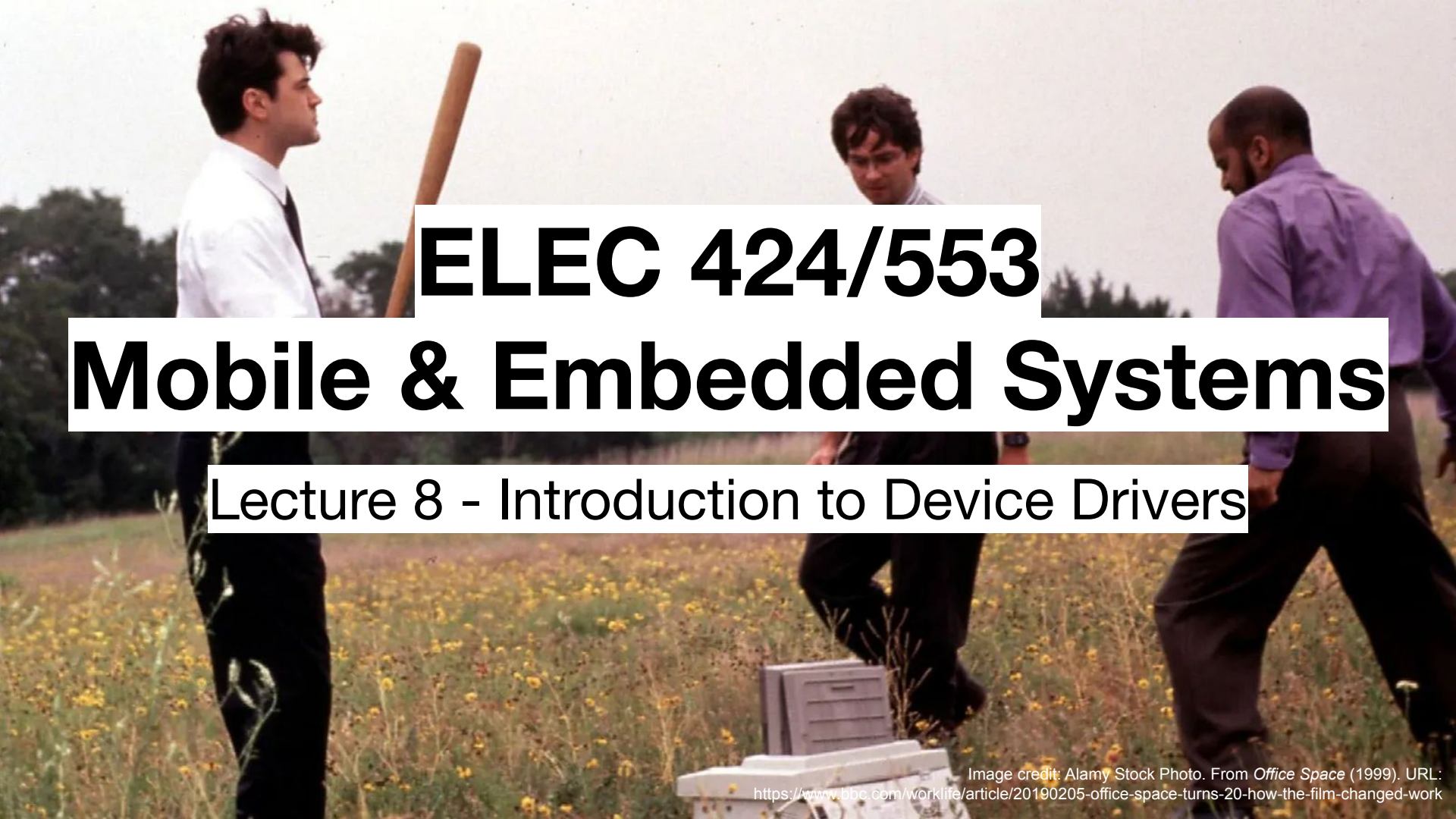


Image credit: Alamy Stock Photo. From *Office Space* (1999). URL: https://www.bbc.com/worklife/article/20190205-office-space-turns-20-how-the-film-changed-work

Questions From Last Class

- Can a process just decide to be on the deadline or realtime schedulers?
  - It depends :)
  - The quoted text below (with my bolding) from: Michael Kerrisk, “sched(7) — Linux manual page”, URL: https://man7.org/linux/man-pages/man7/sched.7.html
    - Look at “Privileges and resource limits”
    - **Deadline**: “A **thread must be privileged** (CAP_SYS_NICE) in order to set or modify a SCHED_DEADLINE policy.”
    - **Realtime**: “the RLIMIT_RTPRIO resource limit **defines a ceiling on an unprivileged thread's static priority** for the SCHED_RR and SCHED_FIFO policies.”
    - **CAP_SYS_NICE**: “Privileged (CAP_SYS_NICE) threads ignore the RLIMIT_RTPRIO limit; as with older kernels, they can make arbitrary changes to scheduling policy and priority.”

Questions From Last Class

CAP_SYS_NICE

* Lower the process nice value (nice(2), setpriority(2)) and change the nice value for arbitrary processes;
* set real-time scheduling policies for calling process, and set scheduling policies and priorities for arbitrary processes (sched_setscheduler(2), sched_setparam(2), sched_setattr(2));
* set CPU affinity for arbitrary processes (sched_setaffinity(2));
* set I/O scheduling class and priority for arbitrary processes (ioprio_set(2));
* apply migrate_pages(2) to arbitrary processes and allow processes to be migrated to arbitrary nodes;
* apply move_pages(2) to arbitrary processes;
* use the **MPOL_MF_MOVE_ALL** flag with mbind(2) and move_pages(2).

Text to the left copy and pasted from:

Michael Kerrisk, “capabilities(7) — Linux manual page”, URL: https://man7.org/linux/man-pages/man7/capabilities.7.html

Questions From Last Class

- What is the ABI? Does it relate to system calls?
  - Application binary interface (ABI)
  - How compiled code should access other compiled code
  - E.g., Compiled application accessing compiled library
- So, for your application to access a system call:
  - For your C source code, use the API (application programming interface) provided by glibc, which will be a function that causes a system call to get executed
  - Your source code gets compiled and will adhere to the ABI for your computer architecture and system configuration
    - In other words, your compiled code will be able to run
  - However, your compiled binary may not execute on another computer with the same architecture (e.g., x86-64) because of differences in system configuration
    - Windows vs. Linux
    - Even within Linux, could be a different distribution, etc.

Questions From Last Class

- Does each core get its own EAX register for system calls?
  - Yes (in x86-64 is it called RAX; EAX is 32-bit)
    - From ENOSUCHBLOG, URL: https://blog.yossarian.net/2020/11/30/How-many-registers-does-an-x86-64-cpu-have
- Pipelining & microkernel - doesn't seem to be much out there
  - https://core.ac.uk/download/pdf/36737603.pdf

Questions From Last Class

- Something I forgot? Let me know!

Housekeeping

- **Assignment 1**
  - Posted soon
- **Soldering Session - Today!**
  - 6:30-8:30 in FE&P 102
  - Last name K and lower in first hour, above K in second hour
  - Required to use safety glasses!!

Soldering Safety

Soldering: users may solder in the lab under the following conditions

- **Approved** soldering irons or desoldering equipment only.
- **Lead** – users should assume that any solder they handle contains lead and take personal precautions accordingly.
- **More than one person** must be present when soldering. No solo work.
- **Do not touch solder** except with hands
- **Wash hands** after handling solder
- **Clean** the work surface after soldering to prevent lead contamination.
- Use **safety glasses** when soldering or sitting next to someone soldering
- **Heat** – don’t touch tip or iron; always put tip in holster when not using
- **Filter** – use the fan + charcoal filters provided when soldering

Housekeeping

- **Office Hours**
  - **Joe**: Tues 3:00-4:00pm in Duncan Hall 2098 or by appointment
  - **Luxi Zhu & Xinbo Luo (TAs)**: Wed 5:30-6:30 in Ryon B10 or by appointment

Device Drivers

- **Software** that enables **interfacing** with **hardware**
- Drivers everywhere
  - Printers
  - Graphics cards
  - Solid state drives (SSDs)
  - Network cards
  - …

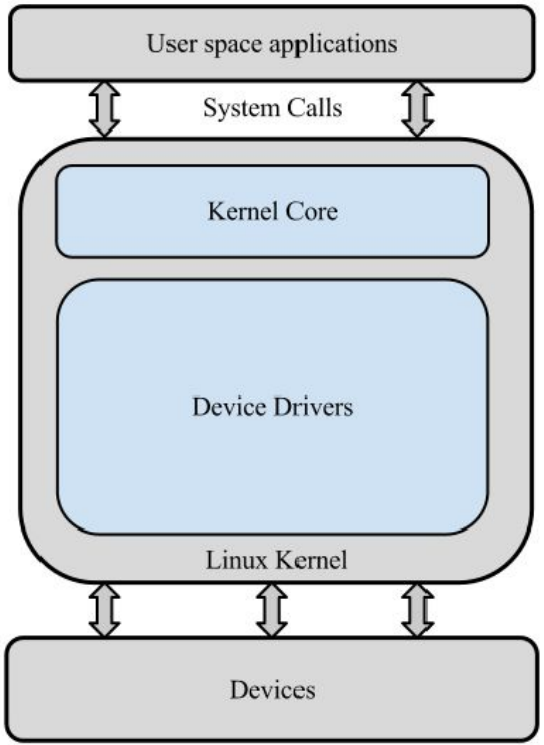


Figure from Zakharov et al, “Generating Environment Model for Linux Device Drivers”. 2013. Conference: 7th Spring/Summer Young Researchers' Colloquium on Software Engineering.
URL: https://www.researchgate.net/publication/301370659_Generating_Environment_Model_for_Linux_Device_Drivers

State of the Art: Linux Drivers

“Most of the code in Linux is device drivers,

Quote from “Device Power Management Basics”.
URL: https://www.kernel.org/doc/html/v5.0/driver-api/pm/devices.html

State of the Art: Linux Drivers

“Most of the code in Linux is device drivers, so most of the Linux power management (PM) code is also driver-specific.

Quote from “Device Power Management Basics”.
URL: https://www.kernel.org/doc/html/v5.0/driver-api/pm/devices.html

State of the Art: Linux Drivers

“Most of the code in Linux is device drivers, so most of the Linux power management (PM) code is also driver-specific. Most drivers will do very little;

Quote from “Device Power Management Basics”.
URL: https://www.kernel.org/doc/html/v5.0/driver-api/pm/devices.html

State of the Art: Linux Drivers

“Most of the code in Linux is device drivers, so most of the Linux power management (PM) code is also driver-specific. Most drivers will do very little; others, especially for platforms with small batteries (like cell phones), will do a lot.”

Quote from “Device Power Management Basics”.
URL: https://www.kernel.org/doc/html/v5.0/driver-api/pm/devices.html

State of the Art: Linux Drivers

“Most of the code in Linux is device drivers, so most of the Linux power management (PM) code is also driver-specific. Most drivers will do very little; others, especially for platforms with small batteries (like cell phones), will do a lot.”

Quote from “Device Power Management Basics”.
URL: https://www.kernel.org/doc/html/v5.0/driver-api/pm/devices.html

State of the Art: Linux Drivers

How much is “most”?

“Most of the code in Linux is device drivers, so most of the Linux power management (PM) code is also driver-specific. Most drivers will do very little; others, especially for platforms with small batteries (like cell phones), will do a lot.”

Quote from “Device Power Management Basics”.
URL: https://www.kernel.org/doc/html/v5.0/driver-api/pm/devices.html

“Talk is cheap. Show me the code.” - Linus Torvalds

“**Device drivers constitute 70% of the Linux code base** [32]

Quote from Kadav & Swift. *ASPLOS’12*, March 3−7, 2012, London, England, UK. (2012) (link)

32: L. Ryzhyk, P. Chubb, I. Kuz, and G. Heiser. Dingo: Taming device drivers. In *Eurosys*. (2009)
14: A. Ganapathi, V. Ganapathi, and D. A. Patterson. Windows xp kernel crash analysis. In *LISA*. (2006)
28: V. Orgovan and M. Tricker. An introduction to driver quality. Microsoft WinHec Presentation DDT301. (2003)

“Talk is cheap. Show me the code.” - Linus Torvalds

“**Device drivers constitute 70% of the Linux code base** [32], and likely are a greater fraction of the code written for the Windows kernel, which supports many more devices.

Quote from Kadav & Swift. *ASPLOS’12*, March 3−7, 2012, London, England, UK. (2012) (link)

32: L. Ryzhyk, P. Chubb, I. Kuz, and G. Heiser. Dingo: Taming device drivers. In *Eurosys*. (2009)
14: A. Ganapathi, V. Ganapathi, and D. A. Patterson. Windows xp kernel crash analysis. In *LISA*. (2006)
28: V. Orgovan and M. Tricker. An introduction to driver quality. Microsoft WinHec Presentation DDT301. (2003)

“Talk is cheap. Show me the code.” - Linus Torvalds

“**Device drivers constitute 70% of the Linux code base** [32], and likely are a greater fraction of the code written for the Windows kernel, which supports many more devices. Several studies have shown that **drivers are the dominant cause of OS crashes** in desktop PCs [14, 28].”

Quote from Kadav & Swift. *ASPLOS’12*, March 3−7, 2012, London, England, UK. (2012) (link)

32: L. Ryzhyk, P. Chubb, I. Kuz, and G. Heiser. Dingo: Taming device drivers. In *Eurosys*. (2009)
14: A. Ganapathi, V. Ganapathi, and D. A. Patterson. Windows xp kernel crash analysis. In *LISA*. (2006)
28: V. Orgovan and M. Tricker. An introduction to driver quality. Microsoft WinHec Presentation DDT301. (2003)

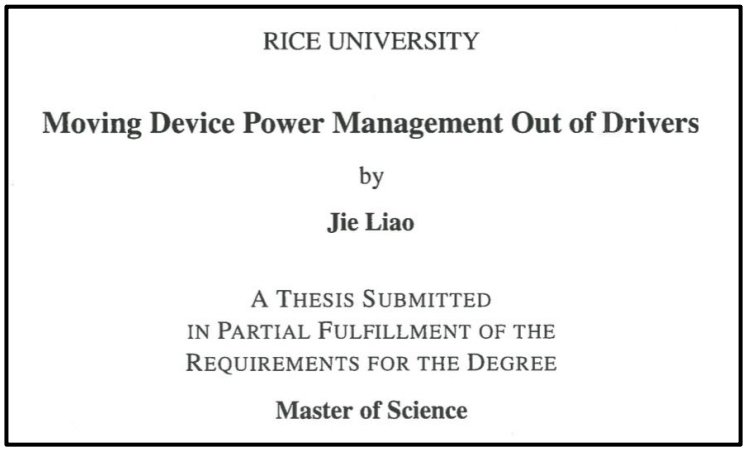
Let’s Hear From Linus Himself

User: Jonsterソ. URL: https://www.youtube.com/watch?v=rF1jLyKIt64 (Language/Content Warning)

Let’s Hear From Linus Himself (Abridged)

User: Jonsterソ. URL: https://www.youtube.com/watch?v=rF1jLyKIt64 (Language/Content Warning)

Can We Simplify Drivers?



RICE UNIVERSITY

Moving Device Power Management Out of Drivers

by

Jie Liao

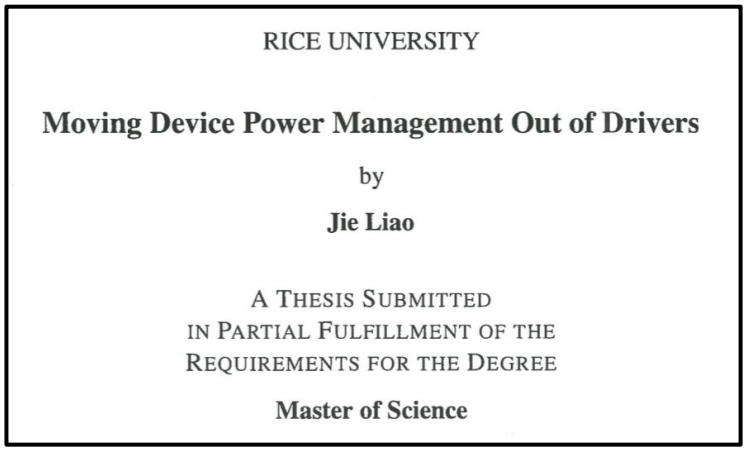
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

Thesis URL: https://scholarship.rice.edu/handle/1911/95976

Can We Simplify Drivers?

“Device drivers are well-known to be complex and error-prone.



RICE UNIVERSITY

Moving Device Power Management Out of Drivers

by

Jie Liao

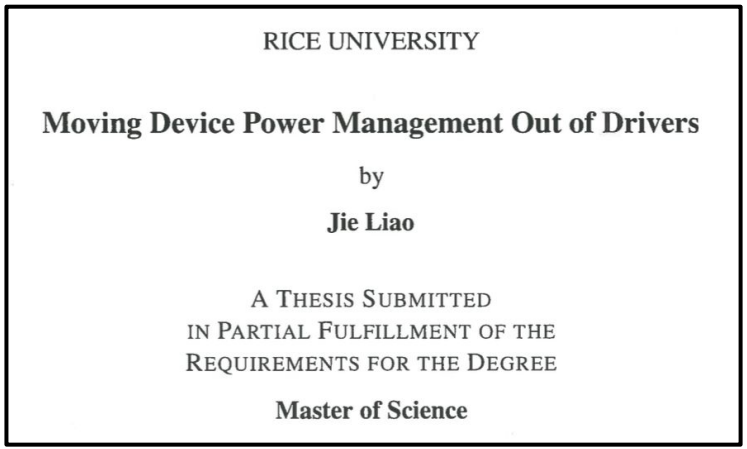
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

Thesis URL: https://scholarship.rice.edu/handle/1911/95976

Can We Simplify Drivers?

“Device drivers are well-known to be complex and error-prone. A widely practiced strategy toward simpler drivers is to move functions out of drivers.



RICE UNIVERSITY

Moving Device Power Management Out of Drivers

by

Jie Liao

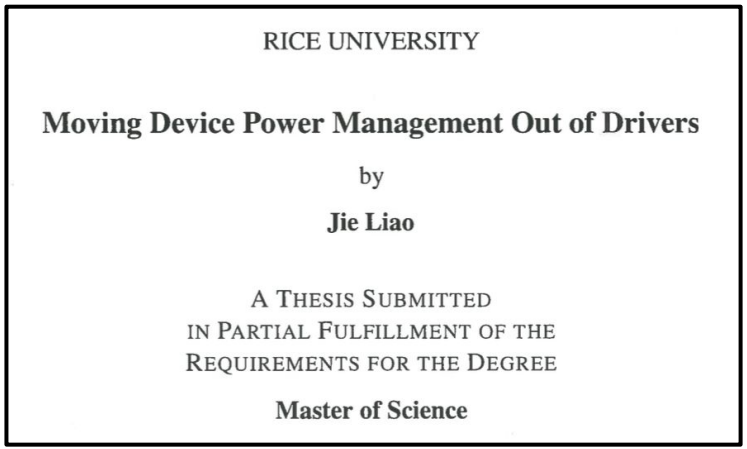
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

Thesis URL: https://scholarship.rice.edu/handle/1911/95976

Can We Simplify Drivers?

“Device drivers are well-known to be complex and error-prone. A widely practiced strategy toward simpler drivers is to move functions out of drivers. Power management (PM), an important function provided by drivers, however, has long resisted this strategy.



RICE UNIVERSITY

Moving Device Power Management Out of Drivers

by

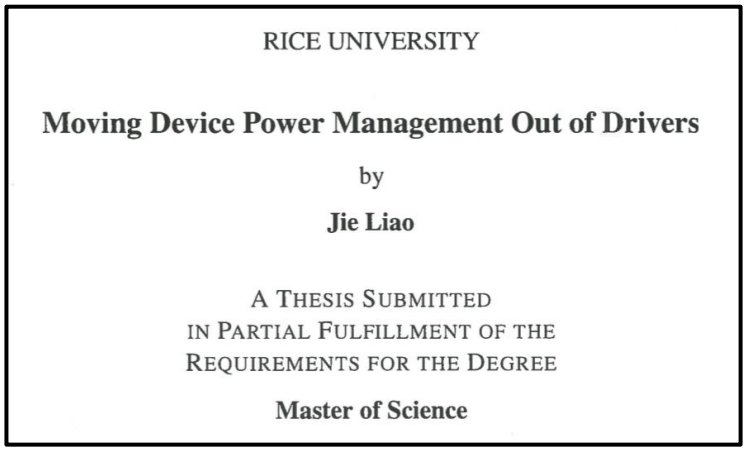
Jie Liao

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

Thesis URL: https://scholarship.rice.edu/handle/1911/95976

Can We Simplify Drivers?



RICE UNIVERSITY

Moving Device Power Management Out of Drivers

by

Jie Liao

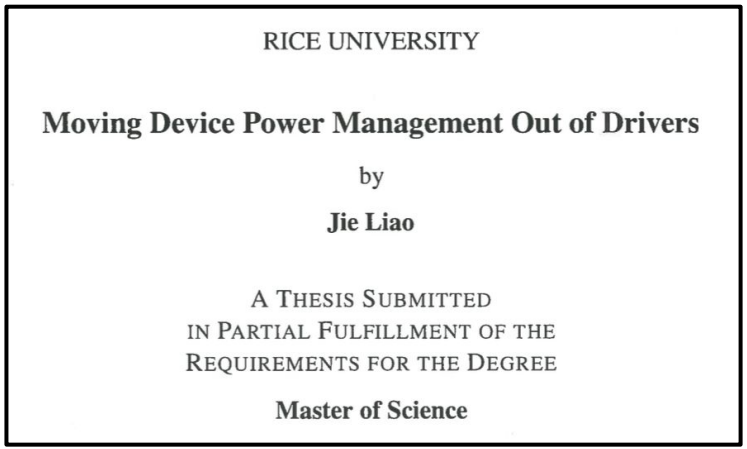
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

"Device drivers are well-known to be complex and error-prone. A widely practiced strategy toward simpler drivers is to move functions out of drivers. Power management (PM), an important function provided by drivers, however, has long resisted this strategy. Today, device drivers provide functions to manage device PM context.

Thesis URL: https://scholarship.rice.edu/handle/1911/95976

Can We Simplify Drivers?



RICE UNIVERSITY

Moving Device Power Management Out of Drivers

by

Jie Liao

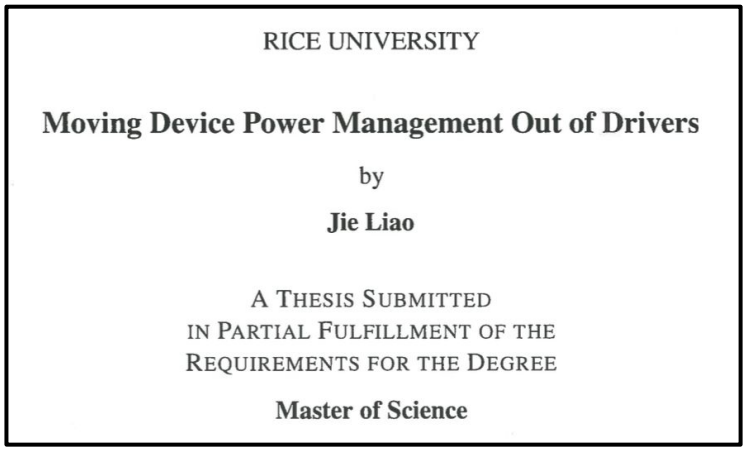
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

"Device drivers are well-known to be complex and error-prone. A widely practiced strategy toward simpler drivers is to move functions out of drivers. Power management (PM), an important function provided by drivers, however, has long resisted this strategy. Today, device drivers provide functions to manage device PM context. This work presents our attempt to move PM context management out of drivers and realize it in a device-independent manner.

Thesis URL: https://scholarship.rice.edu/handle/1911/95976

Can We Simplify Drivers?



RICE UNIVERSITY

Moving Device Power Management Out of Drivers

by

Jie Liao

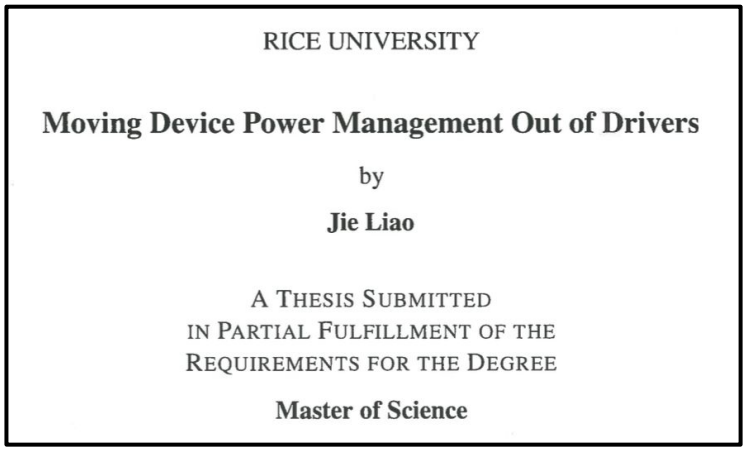
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

"Device drivers are well-known to be complex and error-prone. A widely practiced strategy toward simpler drivers is to move functions out of drivers. Power management (PM), an important function provided by drivers, however, has long resisted this strategy. Today, device drivers provide functions to manage device PM context. This work presents our attempt to move PM context management out of drivers and realize it in a device-independent manner. Our key idea is the rule of representation: separate device-specific knowledge from the logic of PM context management. We design and prototype a generic context manager that provides device-independent suspend/resume logic while allowing device vendors to fold device-specific knowledge into data structures to drive the logic. Our

Thesis URL: https://scholarship.rice.edu/handle/1911/95976

Can We Simplify Drivers?



RICE UNIVERSITY

Moving Device Power Management Out of Drivers

by

Jie Liao

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

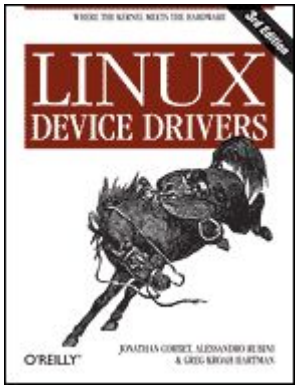
"Device drivers are well-known to be complex and error-prone. A widely practiced strategy toward simpler drivers is to move functions out of drivers. Power management (PM), an important function provided by drivers, however, has long resisted this strategy. Today, device drivers provide functions to manage device PM context. This work presents our attempt to move PM context management out of drivers and realize it in a device-independent manner. Our key idea is the rule of representation: separate device-specific knowledge from the logic of PM context management. We design and prototype a generic context manager that provides device-independent suspend/resume logic while allowing device vendors to fold device-specific knowledge into data structures to drive the logic. Our evaluation on the BeagleBone Black platform shows that our design can move most of suspend/resume code out of drivers, and is as effective as the original driver-based PM context management on power savings." - Abstract of Jie Liao's M.S. thesis

Thesis URL: https://scholarship.rice.edu/handle/1911/95976

Short Stories

Power saving is everything (“electron is king”)
&
GPS+LED init
&
Hour-long discussions

Much of the Following Is Based on Chapter 1 of ...



Linux Device Drivers
Third Edition

Jonathan Corbet
Alessandro Rubini
Greg Kroah-Hartman

Available for free from here: https://lwn.net/Kernel/LDD3/

Kernel Overview

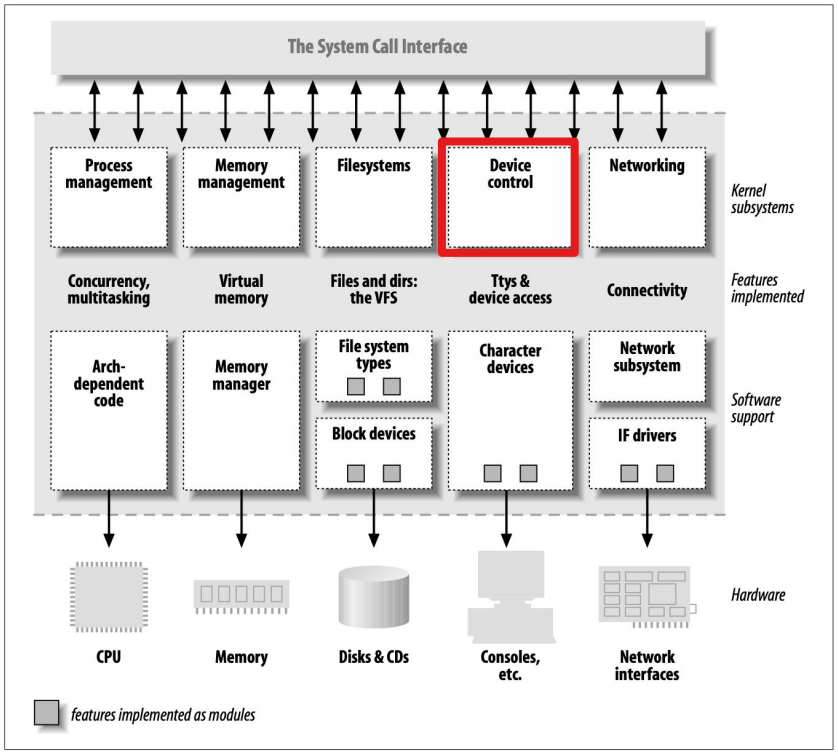
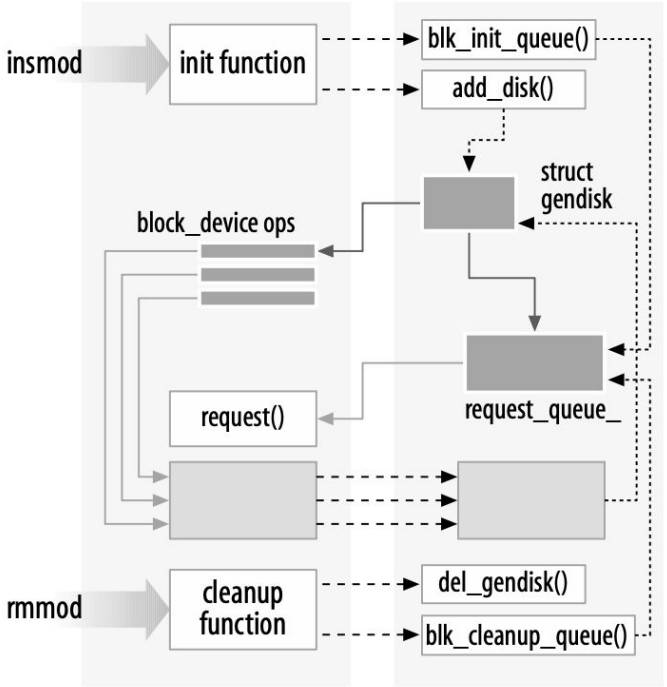


Figure 1-1. A split view of the kernel

Corbet, Rubini, & Kroah-Hartman, *Linux Device Drivers*, 3rd Ed. Fig. from p. 6.

Linux Kernel Modules (LKMs)

- **Loadable** object code
- Enable kernel to do more (or less) on the fly
- Module **classes**
  - E.g. **device drivers**
- Dynamically linked to kernel via **`insmod`**
- Dynamically unlinked to kernel via **`rmmod`**



Corbet, Rubini, & Kroah-Hartman, *Linux Device Drivers*, 3rd Ed. Fig. from p. 19 - full figure not shown.

Modules Used For Different Kernel Services

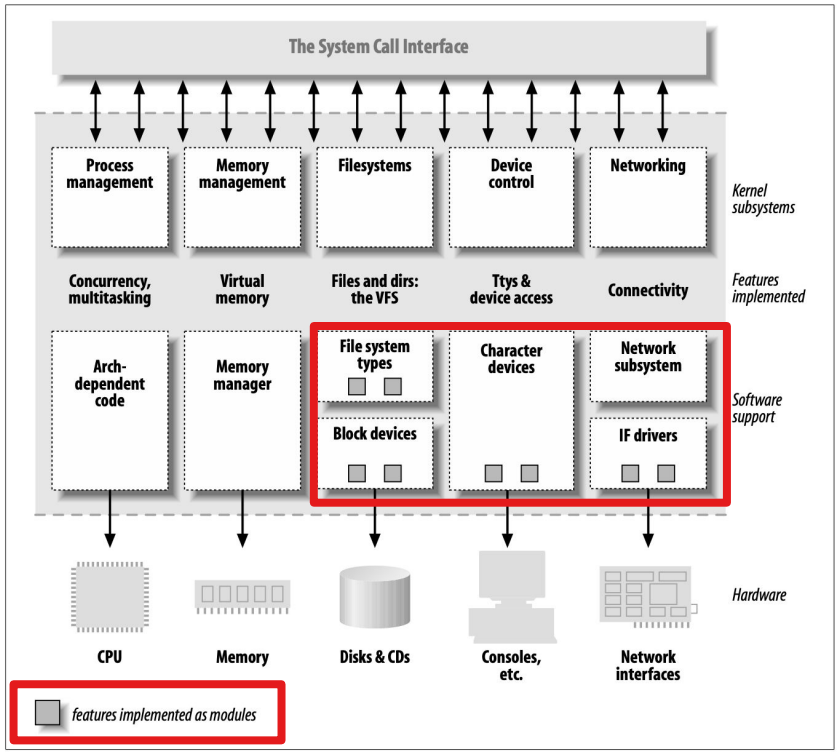


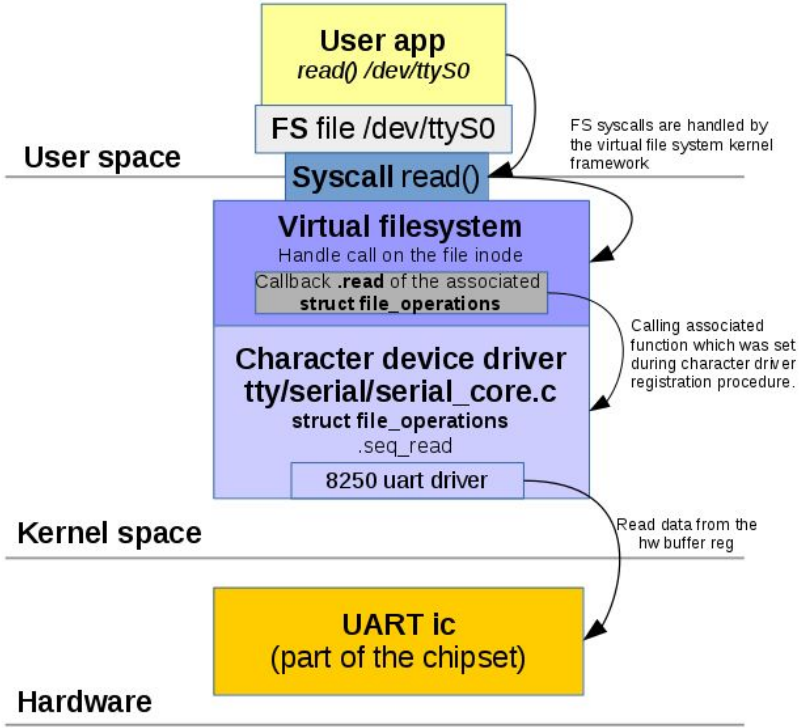
Figure 1-1. A split view of the kernel

- Module class defined by **functionality**
  - **char** module
  - **block** module
  - **network** module

Corbet, Rubini, & Kroah-Hartman, *Linux Device Drivers*, 3rd Ed. Fig. from p. 6.

Character (char) Devices (*cdevs*)

- Interfaced through **stream of bytes**
- System calls commonly used include:
  - `open`
  - `close`
  - `read`
  - `write`
- Similar to **files**
  - But generally limited to sequential access
- Appear in **`/dev`**



Corbet, Rubini, & Kroah-Hartman, *Linux Device Drivers*, 3rd Ed.
Figure: Oleg Kutkov, "Simple Linux character device driver". URL: https://olegkutkov.me/2018/03/14/simple-linux-character-device-driver/

Examples of Character Devices

- Keyboards
- Mice
- Printers

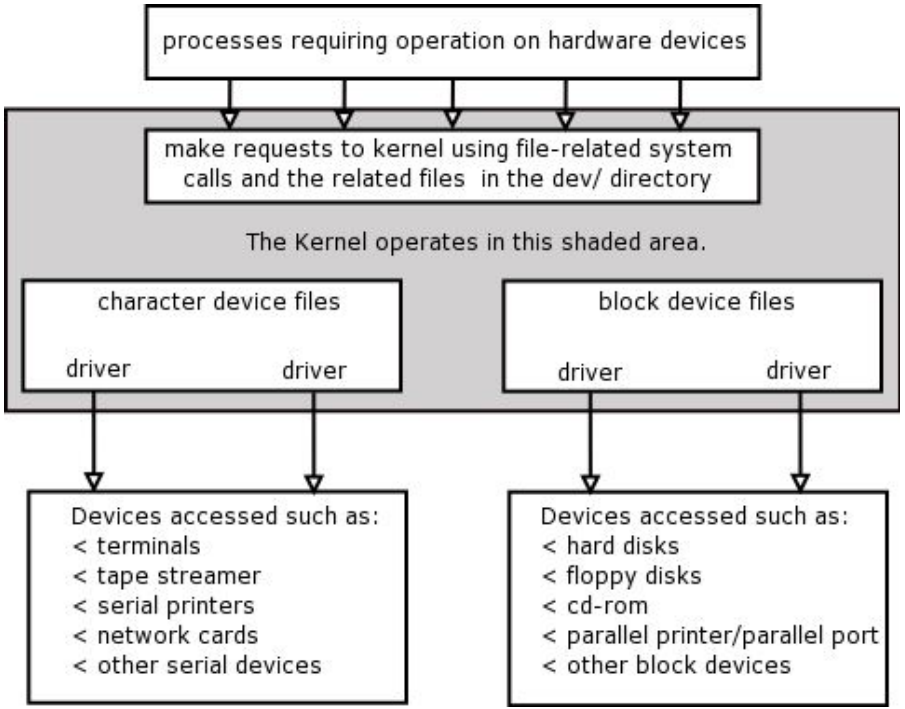
Block Devices (*blkdevs*)

- Device with **filesystem-hosting** capability
  - E.g. solid state drive (SSD)
- Also appear in `/dev`
- Unix flavors generally limited to whole block (~512 bytes) I/O
- Linux can perform arbitrary byte-level I/O
- Applications can access block & char devices the same way
  - Kernel has to figure out how to work with each

Corbet, Rubini, & Kroah-Hartman, *Linux Device Drivers*, 3rd Ed.
Picture: Modified from https://me.me/i/i-missed-the-part-where-thats-my-problem-84a8acd93c234e08990d6d7769fc2a34

Block Devices

- Device with **filesystem-hosting** capability
    - E.g. solid state drive (SSD)
- Also appear in `/dev`
- Unix flavors generally limited to whole block (~512 bytes) I/O
- Linux can perform arbitrary byte-level I/O
- Applications can access block & char devices the same way
    - Kernel has to figure out how to work with each



Corbet, Rubini, & Kroah-Hartman, *Linux Device Drivers*, 3rd Ed.
Figure: EmbLogic's Blog, “Character Driver VS Block Driver”. URL: https://www.emblogic.com/blog/08/character-driver-vs-block-driver/

Examples of Block Devices

- Hard drives
- Solid state drives (SSDs)
- Blu-ray discs
- Flash memory devices

Network Interfaces

- HW/SW devices used to trade **data packets**
  - SW: Loopback interface
- No filesystem entry
- Unix/Linux: Use name like **`eth0`**
  - Remember **`wlan0`** in Project 1?
- **Functions** used by kernel rather than read/write for char & block
  - **Socket API**
  - “Everything is a file” motto of Unix violated

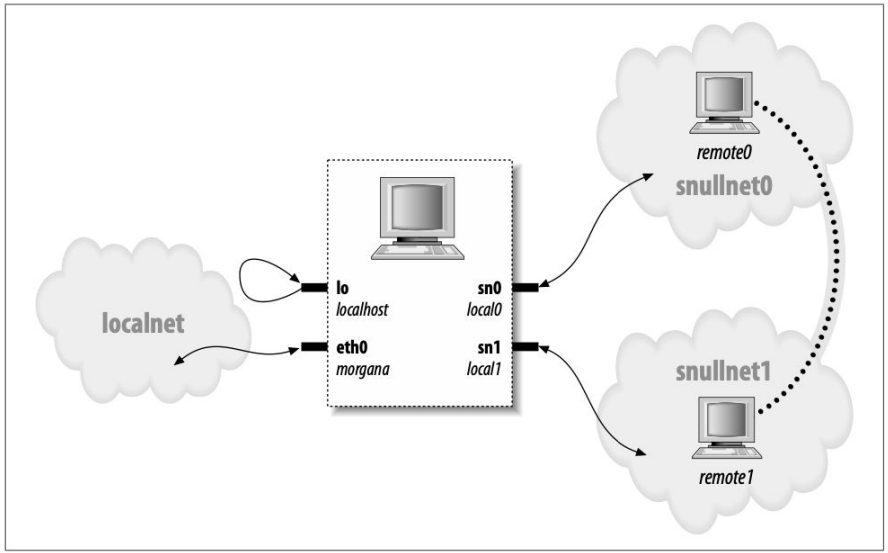


Figure 17-1. How a host sees its interfaces

Corbet, Rubini, & Kroah-Hartman, *Linux Device Drivers*, 3rd Ed. Fig. from p. 500.

Where Do USB Devices Fit In All of This?

- USB devices will appear to Linux as one of the three aforementioned types
- USB serial port
  - Char device
- USB memory card reader
  - Block device
- USB Ethernet interface
  - Network device

Security Motivation

- Should anyone be able to load a module? Why or why not?
- With modules, we're in kernel space - so the answer is a strong "no"
- Verification of process permission to insert modules performed by system call **`init_module`**
  - Hacker goal: Gain permission
- Generally prefer to have higher-level security management
  - Similar to driver power management discussion
- But sometimes, driver writers have to care

Security Considerations

- C will not perfectly protect you
  - Buffer overrun
- Be paranoid about user process input
- Don’t let device receive arbitrary information from user
- Enforce user access levels

‘How to Solve a Buffer Overrun Detected Problem’. URL: https://www.tech-faq.com/how-to-solve-buffer-overrun-detected-problem.html

How Does GPL Relate to Drivers & Modules?

- Linux has GPLv2
- “The GPL allows anybody to redistribute, and even sell, a product covered by the GPL, as long as the recipient has access to the source and is able to exercise the same rights.” - LDD
- “Additionally, any software **product derived from a product covered by the GPL must**, if it is redistributed at all, **be released under the GPL**.” - LDD
- Are drivers a derived product?
- Can vendors give just binaries of their modules/drivers?

Slide based on and quotes from: Corbet, Rubini, & Kroah-Hartman, *Linux Device Drivers*, 3rd Ed.

How Does GPL Relate to Drivers & Modules?

“1. **You may copy and distribute verbatim copies of the Program's source code** as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.”

“GNU General Public License, version 2”. URL: https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html. Bolding is my own.

How Does GPL Relate to Drivers & Modules?

"3. **You may copy and distribute the Program** (or a work based on it, under Section 2) **in object code** or executable form under the terms of Sections 1 and 2 above **provided that you also do one of the following**:

a) **Accompany it with the complete corresponding machine-readable source code**, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, **a complete machine-readable copy of the corresponding source code**, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) **Accompany it with the information you received as to the offer to distribute corresponding source code**. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it."

"GNU General Public License, version 2". URL: https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html

How Does GPL Relate to Drivers & Modules?

- Up to this point, it's been okay to only give module binaries
- Overall message - modules not considered derived products
- Tread cautiously - not uniform consensus among developers
- "Talk to legal"
- Of course, this is different for mainline kernel modules
    - Those require a GPL-compatible license
    - You must give out the source!

?

Slide based on and quotes from: Corbet, Rubini, & Kroah-Hartman, *Linux Device Drivers*, 3rd Ed.
Image URL: https://www.hunts.com/tomato-sauce-and-paste/tomato-sauce

Next Time

Writing Your Very Own Kernel Module