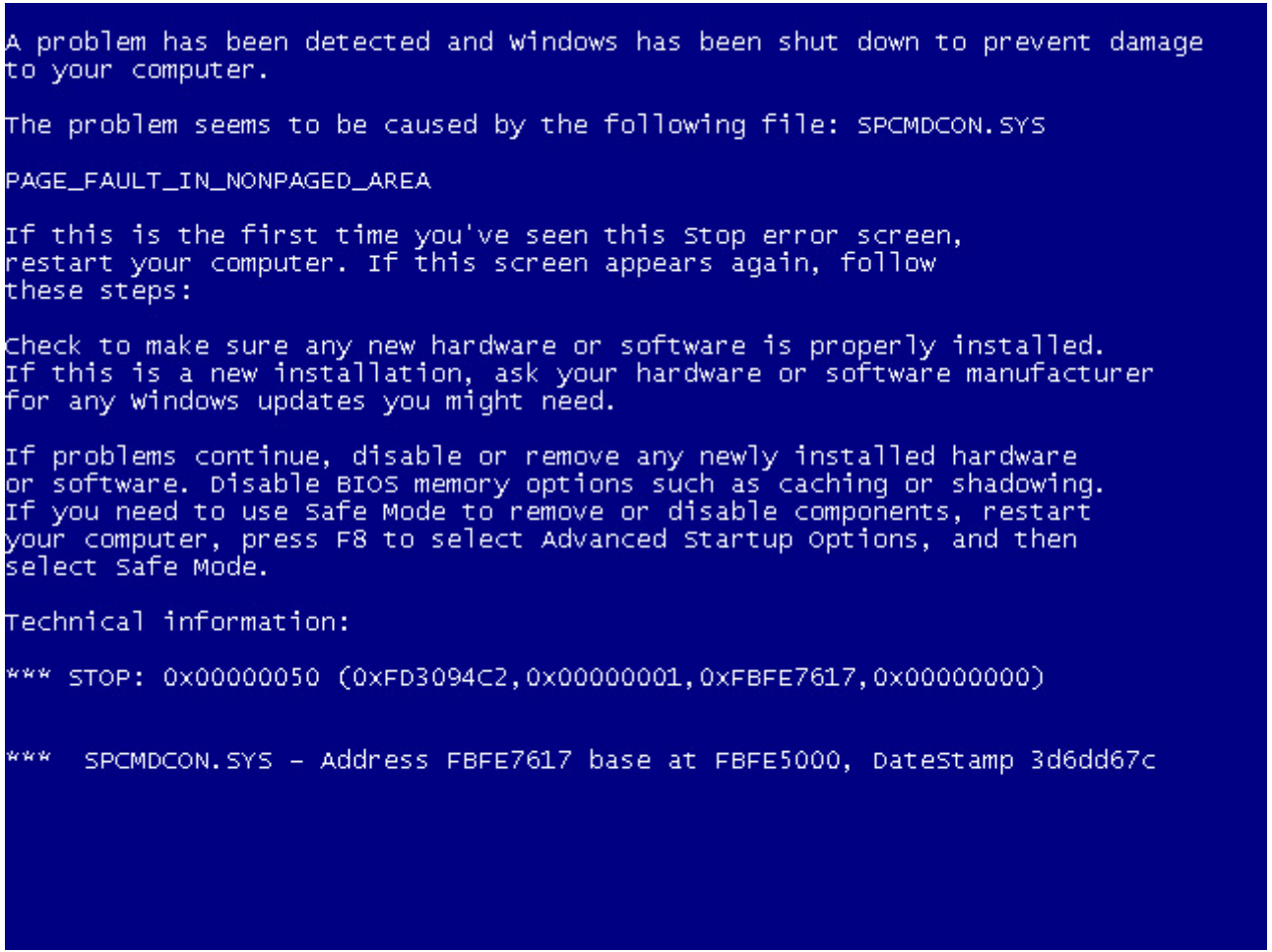
Debugging im Linux Kernel

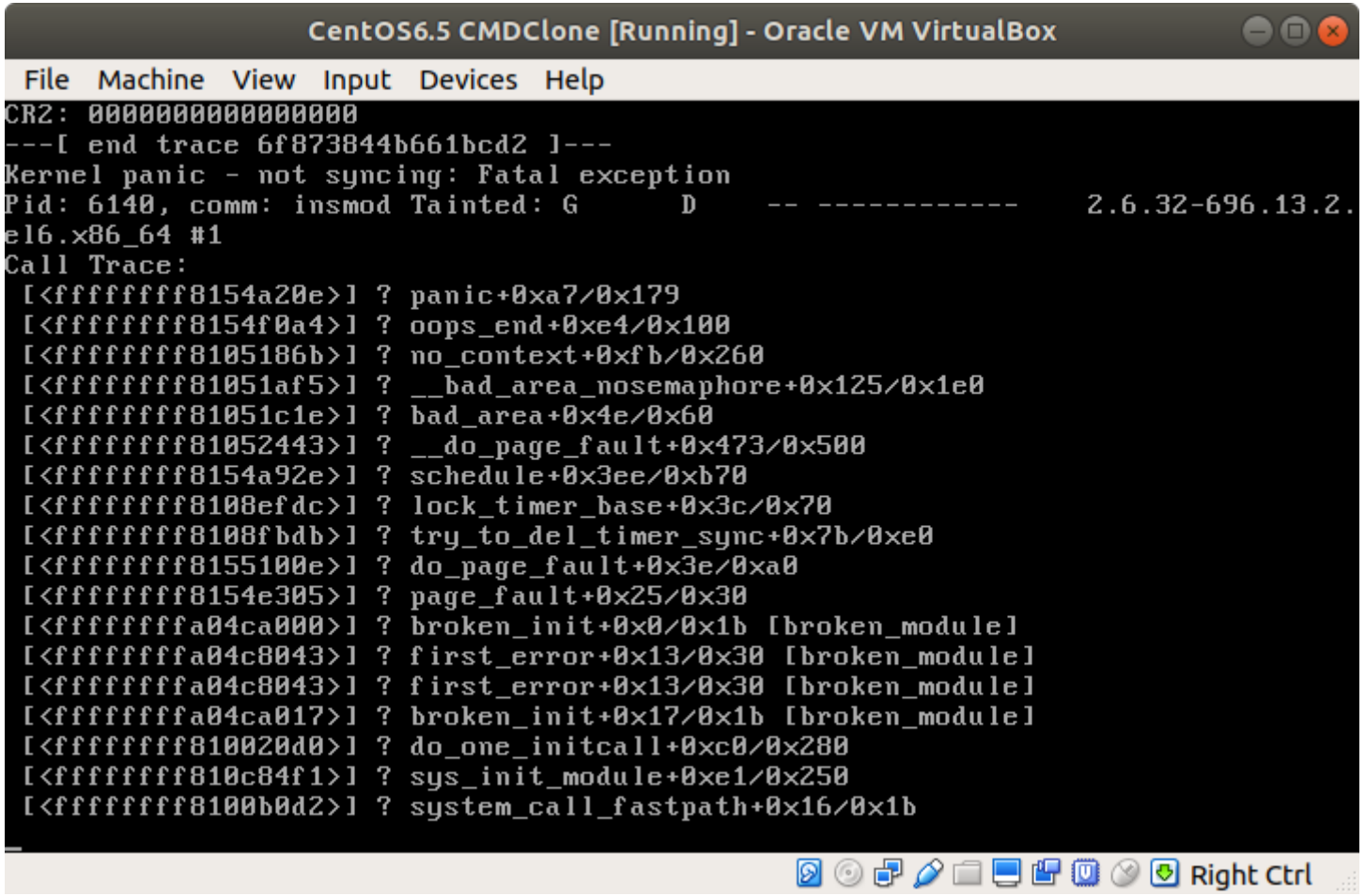
Seminar Effiziente Programmierung:
14.12.17 Marc David

Arbeitsbereich Wissenschaftliches Rechnen Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

Blue Screen [1]



Kernel Panic



Gliederung

- Debugging von Normalen Programmen
- Linux Kernel Grundlagen
- Kernel Oops, Kernel Panic
- Tools
- Demo
- Weitere Tools
- Zusammenfassung

Debugging Definition

- Debugging is the process of finding and resolving defects or problems within the program that prevent correct operation of computer software or a system. [2]

- Reproduzierbarkeit

Verfahren zum Debuggen von Programmen (Auswahl)

- Code/Fehlermeldung lesen
- Print Statements
- Debugger (gdb, pdb, JTAG)
- Logs/Dumps
- Testing (z. B. Unit Tests)
- Strace
- etc.

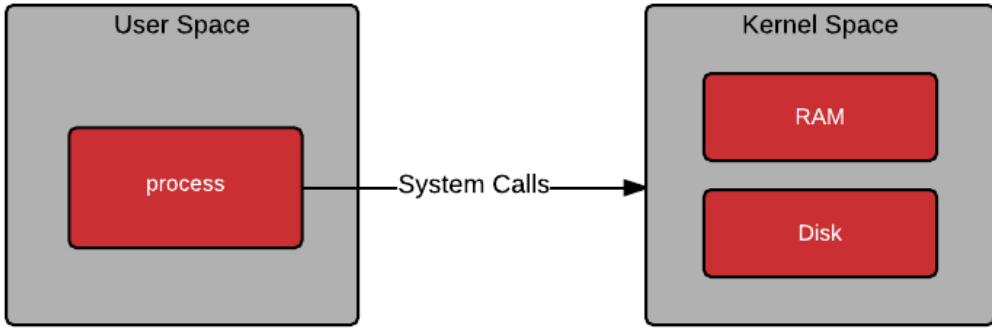
Betriebssysteme

- Abstraktion von der Hardware
  - CPU, RAM, Netzwerk etc.

- Ressourcenverteilung
  - Verwaltung von Betriebesmitteln (Ram. I/O, CPU-Zeit etc.)

User-Space Kernel Space

- User Space
  - Code der außerhalb des Kernels arbeitet (ls, cat etc.)
  - Begrenzte Rechte
- Kernel Space
  - Code innerhalb des Kernels
  - Volle Rechte
- System Calls
  - Schnittstelle des Kernels für normale Programme



[3]

Linux Kernel Module

- In C geschrieben
  - Anderes Makefile als normales C Programm
- init und exit Funktionen
  - `module_init(broken_init);`
  - `module_exit(broken_exit);`
- Keine Sequentielle Ausführung (Anmeldung für Event)
- Einfügen in den Kernel (zur Laufzeit):
  - `Insmod, modprobe`
- `lsmod, modinfo, rmmod`

Kernel Oops

- Kernel killt Prozess
- Kann Zuverlässigkeit des Systems beeinflussen
  - Reboot empfohlen
- Fehlermeldung wird geloggt
  - Kurze Beschreibung, Call Trace, Registerinformationen
- Gründe:
  - Code im Kernel möchte zu viel Speicher alloziert bekommen
  - Überschreiben von fremden Speicher (ggf.)

Kernel Panic

- Aufruf der Funktion panic() innerhalb des Kernels
  - Alle Prozesse werden gestoppt
  - Informationen werden auf der Konsole ggf. ausgegeben
- System muss neu gestartet werden.
- Ursachen:
  - Hardware Probleme
  - Null Pointer Dereference
  - Doppeltes Free

Printing

- Userspace Print: Ausgabe auf Konsole
- `printk(KERN_INFO "Hallo Welt");`
- Loglevel: einfacher String
  - 7 Stufen von KERN_EMERG bis KERN_DEBUG
- *Zuerst in System-Buffer gespeichert*
  - */proc*/kmsg
- *Speicherung /var/log/messages (Centos)*
  - *asynchron*
- *Ausgabe auf der Konsole möglich*

Kernel-Panic: Kdump

- Tool um Dump im Falle einer Kernel Panic zu sammeln
- Separater Kernel
  - Reservierter Speicherbereich
- Im Fall einer Panic:
  - Booten des seperaten Kernel (kexec)
  - erstellt Speicherabbild (oft komprimiert)
  - Default Speicherort “/var/crash”

Kernel-Panic: Crash

- Tool zur Analyse des Crashdumps (Post mortem Analyse)
  - GDB kann keine 64bit ELF Dateien lesen [4]
- Vmlinux:
  - Kompilierter Linux Kernel mit Debugging Informationen (-g)
  - Gleiche/ähnliche Kernel Version nötig
- Vmcore: Dump der von Kdump erstellt wurde
- Log, Backtrace, Disassembler

Demo

Demo Kernel Panic

- Quellcode Null Pointer Dereference
  - User Space
- Quellcode Null Pointer Dereference
  - Kernel Modul
- Insmod Modul im Kernel einfügen

- Crash Dump inspizieren
  - Log, bt, dis,

Weitere Fehler

- C Speicherverwaltung
  - `(k)malloc` (Rückgabe Pointer mit Beginn des allozierten Speichers)
  - `Free` ()
- Doppeltes Free (Kernel Panic)
- Viel zu viel Speicher allozieren (Kernel Oops)
- Drüber hinaus schreiben von allozierten Speicher
- Weiterschreiben auf Speicher nach free-Aufruf
- Zugriff auf zufällige Adressen

GDB

- Kommandozeilen Debugger
- Arbeitet im Userspace
- „-g“ Option von GCC wird benötigt
  - Debugging Symbole

- Kernel Space nicht so einfach:
  - z. B. Nutzereingaben müssen verarbeitet werden
  - viele Prozesse laufen gleichzeitig

KDB/KGDB

- Debugger für den Linux Kernel
- Option wenn man sich den Kernel kompiliert
- Zwei Linux Maschinen benötigt
  - Verbunden über serielle Schnittstellen
- KDB Backtrace für aktuellen Prozess, Kernelmodule anzeigen etc.
- KGDB (Remote GDB Session)

Zusammenfassung

- Debugging im Kernel ist aufwändiger, schwieriger
- Kernel Space/User Space
- Zwei Arten von Fehlern
  - Kernel Panic
  - Kernel Oops
- Tools
  - Kdump, Crash, KDB/KGDB

Quellen

- [1] Bluescreen Windows https://upload.wikimedia.org/wikipedia/commons/a/a8/Windows_XP_BSOD.png Eingesehen am 01.12.2017
- [2] „Debugging Definition Wikipedia“ https://en.wikipedia.org/wiki/Debugging, Eingesehen am 01.12.2017
- [3] User Space Kernel Space Grafik https://rhelblog.files.wordpress.com/2015/07/user-space-vs-kernel-space-simple-user-space.png Eingesehen am 8.12.17
- [4] Linus Torvalds über Debugger https://lwn.net/2000/0914/a/lt-debugger.php3 Eingesehen am 01.12.2017
- [5] Kdump Dokumentation: Analysis https://www.kernel.org/doc/Documentation/kdump/kdump.txtEingesehen am 01.12.17

Weitere Quellen - 1

- Andrew S. Tanenbaum: Modern Operating Systems (3rd Edition)
- Linux kernel debugging for sysadmins, Minto Joseph FrOSCon 2017, Eingesehen am 8.12.17: https://media.ccc.de/v/froscon2017-1925-linux_kernel_debugging_for_sysadmins#video
- „You can be a kernel hacker!“, Eingesehen am 8.12.17: https://jvns.ca/blog/2014/09/18/you-can-be-a-kernel-hacker/ Linux Device Drivers, (3rd Edition), Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman (Chapter 2, 4 8 https://lwn.net/Kernel/LDD3/
- The Linux Kernel Module Programming Guide , Eingesehen am 8.12.17: http://www.tldp.org/LDP/lkmpg/2.6/html/x427.html

Weitere Quellen - 2

- Redhat Enterprise Linux Deployment Guide: Chapter 32, Eingesehen am 8.12.17: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/ch-kdump
- White Paper: Red Hat Crash Utility, Eingesehen am 8.12.17: https://people.redhat.com/~anderson/crash_whitepaper/
- Documentation for Kdump - The kexec-based Crash Dumping Solution, Eingesehen am 8.12.17: https://www.kernel.org/doc/Documentation/kdump/kdump.txt
- Decoding the Linux kernel's page allocation failure messages, Eingesehen am 8.12.17: https://utcc.utoronto.ca/~cks/space/blog/linux/DecodingPageAllocFailures

Weitere Quellen - 3

- Using kgdb / gdb, Eingesehen am 8.12.17: https://www.kernel.org/doc/html/v4.13/dev-tools/kgdb.html#using-kgdb-gdb
- Using Modprobe, Eingesehen am 8.12.17: https://www.linuxforums.org/forum/applications/194343-modprobe-fatal-module-hello-ko-not-found.html
- Architecting Containers Part 1:Why Understanding User Space vs. Kernel Space Matters, Eingesehen am 8.12.17: http://rhelblog.redhat.com/2015/07/29/architecting-containers-part-1-user-space-vs-kernel-space/
- Debuggen mit GDB, Eingesehen am 8.12.17: http://openbook.rheinwerk-verlag.de/linux_unix_programmierung/Kap17-005.htm#RxxKap17005040061E1F02B100

Weitere Quellen - 4

- Options for Debugging Your Program, Eingesehen am 8.12.17: https://gcc.gnu.org/onlinedocs/gcc/Debugging-Options.html
- Professional Linux kernel architecture Wolfgang Mauerer (2008)
- Kernel Parameter für Linux, Eingesehen am 8.12.17: https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt
- Analyzing Linux kernel crash dumps with crash, Eingesehen am 8.12.17: https://www.dedoimedo.com/computers/crash-analyze.html
- Linux Kernel: memory corruption - debug tricks, Eingesehen am 8.12.17: https://helenfornazier.blogspot.de/2015/07/linux-kernel-memory-corruption-debug.html

Weitere Quellen 5

- Writing a Linux Kernel Module — Part 1: Introduction
  http://derekmolloy.ie/writing-a-linux-kernel-module-part-1-introduction/