System Monitoring mit strace

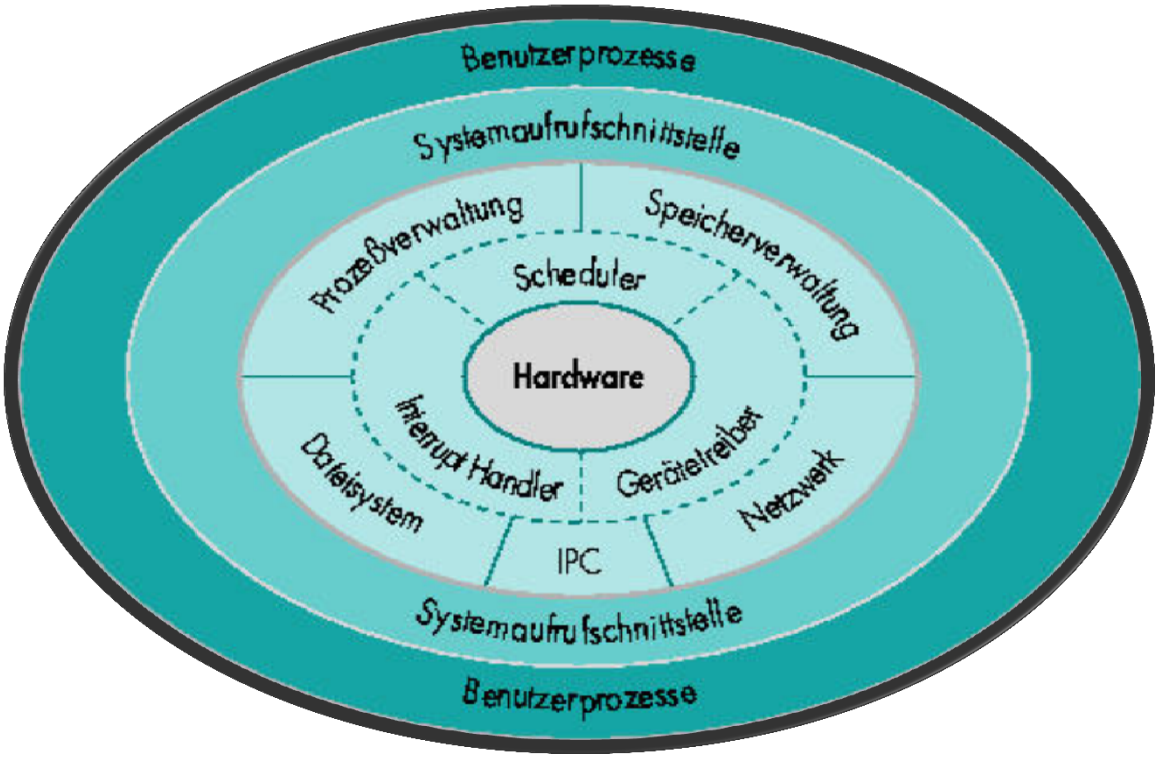
Systemcall tracing

Gliederung

- Einleitung: Was ist strace
- Grundlagen zu strace
  - Kernel
  - Kernelspace vs. Userspace
  - Systemcalls
  - ptrace
- Simple strace (Demo)
- strace die wichtigsten Funktionen

Kurze Übersicht: Was ist strace

- Der Name kommt von: Systemcall Tracing
- Systemcalls sind Aufrufe an den Kernel
- Mit strace lasses sich die Systemcall überwachen



Kurze Übersicht: Was ist strace

- Überwachen von Systemcalls bedeutet:
  - Protokollierung:
    - der Namen der aufgerufenen Systemcalls
    - der Parameter die übergeben werden
    - der Rückgabewerte die geliefert werden
  - Typischer Anwendungsfall
    - Ein Programm startet nicht und gibt als Fehler an: „Fehler 5“

Kurze Übersicht: Was ist strace

- strace Beispiel Output:

```
execve("./hworld", ["./hworld"], [/* 40 vars */]) = 0
brk(0)                                  = 0x1b6a000
access("/etc/ld.so.nohwcap", F_OK)  = -1 ENOENT (No such file or
  directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
  -1, 0) = 0x7f40782d4000
access("/etc/ld.so.preload", R_OK)  = -1 ENOENT (No such file or
  directory)
open("/etc/ld.so.cache", O_RDONLY)  = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=107986, ...}) = 0
mmap(NULL, 107986, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f40782b9000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)  = -1 ENOENT (No such file or
  directory)
open("/lib/libc.so.6", O_RDONLY)    = 3
```

Grundlagen zu strace

Kernel / Systemcalls / ptrace

Grundlagen: Der Kernel

- Der Kernel, immer da und nie zusehen.
- Was macht der Kernel:
  - Abstraktion der Hardware
  - Prozessverwaltung
  - Speicherverwaltung
  - Netzwerk
  - Dateisystem
  - Inter-process communication (IPC)

Grundlagen: Der Kernel

- Wie macht der Kernel das?

- Monolithischer Kernel

  - Der Kernel selbst regelt alle nötigen Aufgaben
  - Es gibt keine zusätzlichen Dienste die Kernelaufgaben übernehmen

Grundlagen: Der Kernel

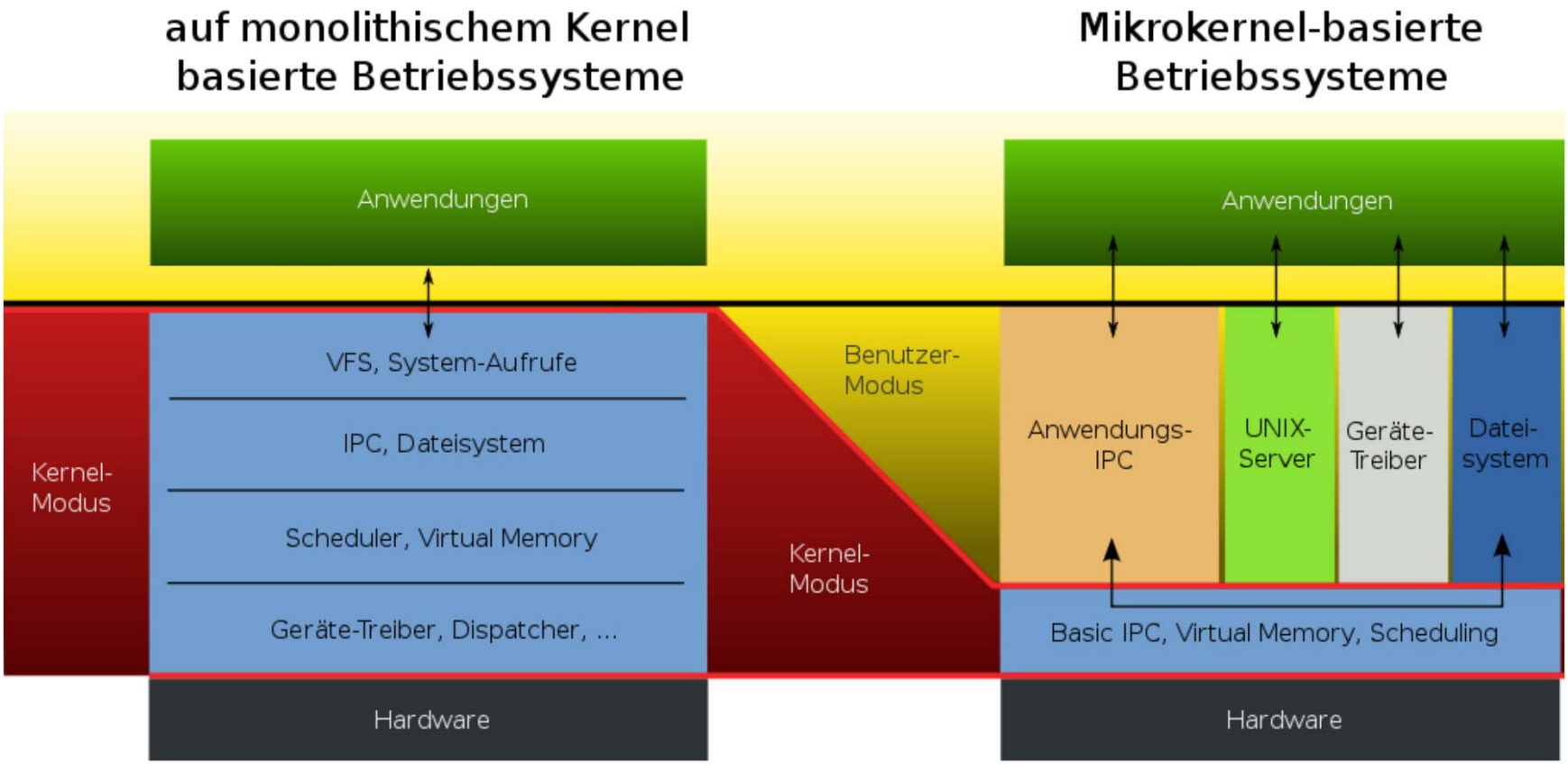
- Wie macht der Kernel das?

- Mikrokernel

  - Der Kernel beschränkt sich auf die grundlegendsten Aufgaben
    - Prozessverwaltung
    - Speicherverwaltung

  - Andere Aufgaben werden in einer höheren Schicht, als Dienst realisiert
    - Dateisystem
    - Netzwerk

Grundlagen: Der Kernel



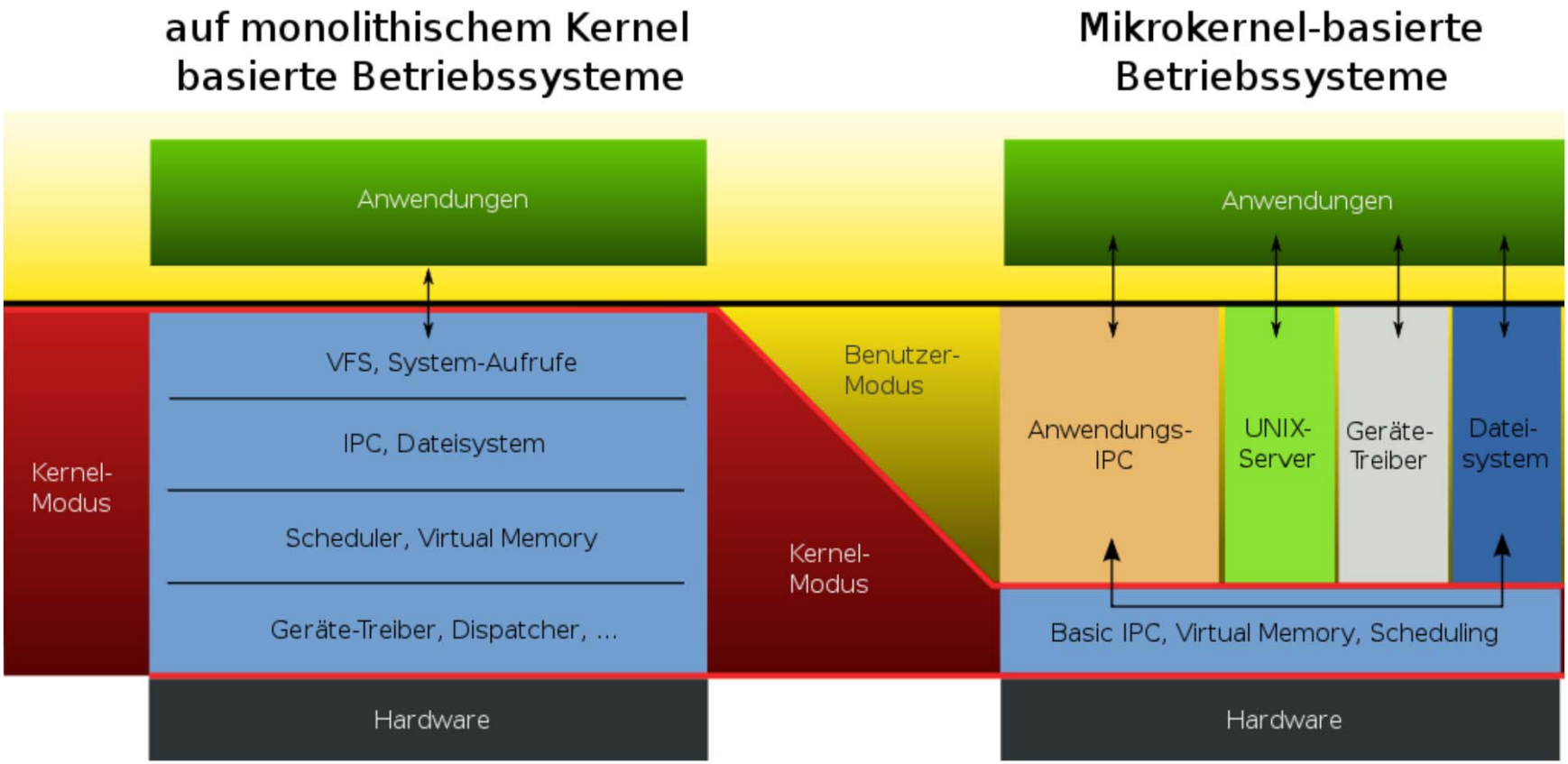
Grundlagen: Der Kernel

- Mikrokernel
  - Pro:
    - Modular aufgebaut, gutes Software Design
    - Sicherer gegen Abstürze
  - Kontra:
    - Langsam
- Monolithischer Kernel
  - Pro:
    - Schnell
  - Kontra:
    - Code ist unübersichtlicher
    - Etwas unsicherer gegen Abstürze

Grundlagen: Der Kernel

- Userspace und Kernelspace
  - In Linux Systemen wird der Speicher in zwei Bereiche getrennt
  - Den **Kernelspace:**
    - Hier laufen die Service die der Kernel bereitstellt.
    - Ins Besondere die Systemcalls
  - Den **Userspace:**
    - Hier laufen alle Userprozesse
    - d.h. alles außer dem Kernel

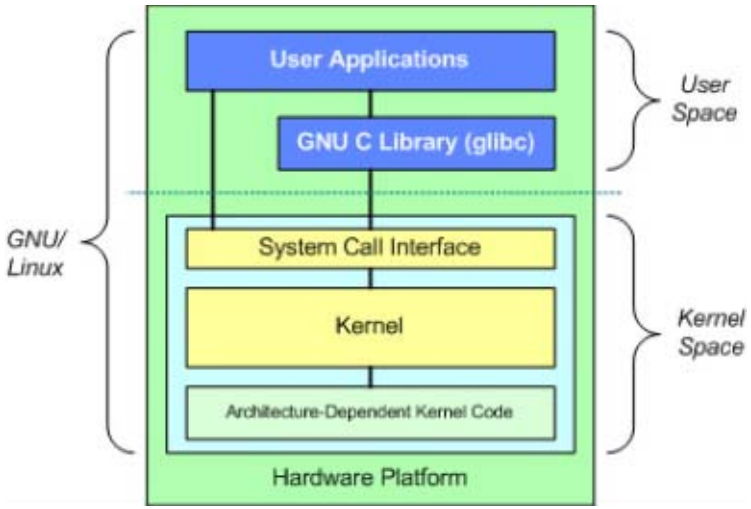
Grundlagen: Der Kernel



Grundlagen: Systemcalls

- Systemcalls sind Aufrufe an den Kernel
  - Datei öffnen, Speicher anfordern, auf den Bildschirm schreiben
- Sie bilden die Brücke zwischen Kernelspace und Userspace
- Sie sind die einzige Möglichkeit mit dem Kernel zu kommunizieren

Grundlagen: Systemcalls



Grundlagen: Systemcalls

- Hat jemand schon mal einen gesehen?

- Systemcalls werden von den Standardbibliotheken der Programmiersprachen gekapselt.
  - Java: `System.out.print(„Hello World“);`
  - C: `printf(„Hello World“);`

  - Systemcall (in C):

```
char buf[] = „Hello World“;
syscall(4, 1, buf, 11);
```

Grundlagen: Systemcalls

- Systemcall „write“ in Assembler (32bit)

```
section .data
  s:      db 'Hello world',10
  sLen:   equ $-s

_start:
  mov eax, 4            ; Nr. des syscall
  mov ebx, 1            ; Nr. des Outputs (1 = stdout)
  mov ecx, s;           ; der String
  mov edx, sLen;        ; Länge des Strings

  int 80h               ; löst den Interrupt aus
```

Grundlagen: ptrace

- Ein Systemcall im Speziellen: ptrace
- strace basiert wesentlich auf ptrace
- Möglichkeiten mit ptrace:
  - Auslesen von Werten im Adressraum des Prozesses
  - Manipulieren von Werten im Adressraum des Prozesses

Grundlagen: ptrace

- Einsatzgebiet:
  - Tracing Tools wie strace
    - Anhängen an Prozesse
    - Zyklisches auslesen des Prozessadressraums
  - Debugger wie GDB
    - Breakpoint Debugging
    - Single Step Debugging
  - Code Injection
    - positive wie negative

Grundlagen: ptrace

- Aufruf: `ptrace(opt, pid, addr, data)`
- ptrace die wichtigsten Optionen:

- `PTRACE_ATTACH, PTRACE_DETACH`
  - Aktiviert/deaktiviert die Verfolgung eines Prozesses. Der Prozess wird bei jedem Signal unterbrochen und der Verfolger informiert.

- `PEEKUSR, PEEKTEXT, PEEKDATA`
  - Daten den aus den Registern, dem Textbereich, dem Datenbereich können ausgelesen werden

Grundlagen: ptrace

- ptrace die wichtigsten Optionen:

- `POKEUSR, POKETEXT, POKEDATA`
  - Daten in die Register, dem Textbereich, den Datenbereich schreiben

- `PTRACE_SYSCALL`
  - Der Kern startet die Ausführung des Prozesses, er stoppt ihn bevor und nachdem ein Systemcall ausgeführt wird.

- `PTRACE_SINGLESTEP`
  - Im Singlestep Modus unterbricht der Kern, den Prozess, nach jeder Assembler-Anweisung

Grundlagen: ptrace

- Implementation von sstrace (simple strace)

- Drei wesenliche Teile
  1. Dem Kernel mitteilen das der Prozess verfolgt werden soll (attach)
  2. Anhängen an den (jeweils) nächsten Systemcall
  3. Auslesen der Informationen

Grundlagen: ptrace

1. Dem Kernel mitteilen das der Prozess verfolgt werden soll

   a. Die Signale vom Childprozess abfangen. (Einen Listener einhängen)

   b. Den Prozess <pid> zum Child Prozess machen. Das eigentliche „Anhängen“, der Prozess <pid> wird zum Childprozess des tracing Programmes

```
ptrace(PTRACE_ATTACH, <pid>, 0, 0);
```

Grundlagen: ptrace

2. Anhängen an den (jeweils) nächsten Systemcall

   a. Der Prozess <pid> soll vor und nach jedem Systemcall unterbrechen, und ein Signal an den Parentprozess schicken

```
ptrace(PTRACE_SYSCALL,<pid>,(char*)1,0)
```

Grundlagen: ptrace

3. Auslesen der Informationen

a. Die, in den Registern stehenden Werte werden ausgelesen.

```
val = ptrace(PTRACE_PEEKUSER,pid,off,0)
```

Grundlagen: ptrace

Demo

Simple Strace (sstrace)

strace

Die wichtigsten Optionen

strace

- Aufrufe:
  - `strace –p <pid> [-o outfile]`
  - `strace [-o outfile] command`

strace

- Optionen:

-f Traced die auch entstandene Childprozesse

-i gibt den Instruktions-Pointer mit aus (Adresse)

-e Filtert die Ausgabe. Wird „**file**“ übergeben werden nur Systemcalls die mit files zusammenhängen getraced.

-e file

-e trace=open

-e open

-e read=open

strace

- Optionen:

-c gibt eine Statistik über die aufgerufenen Systemcalls aus

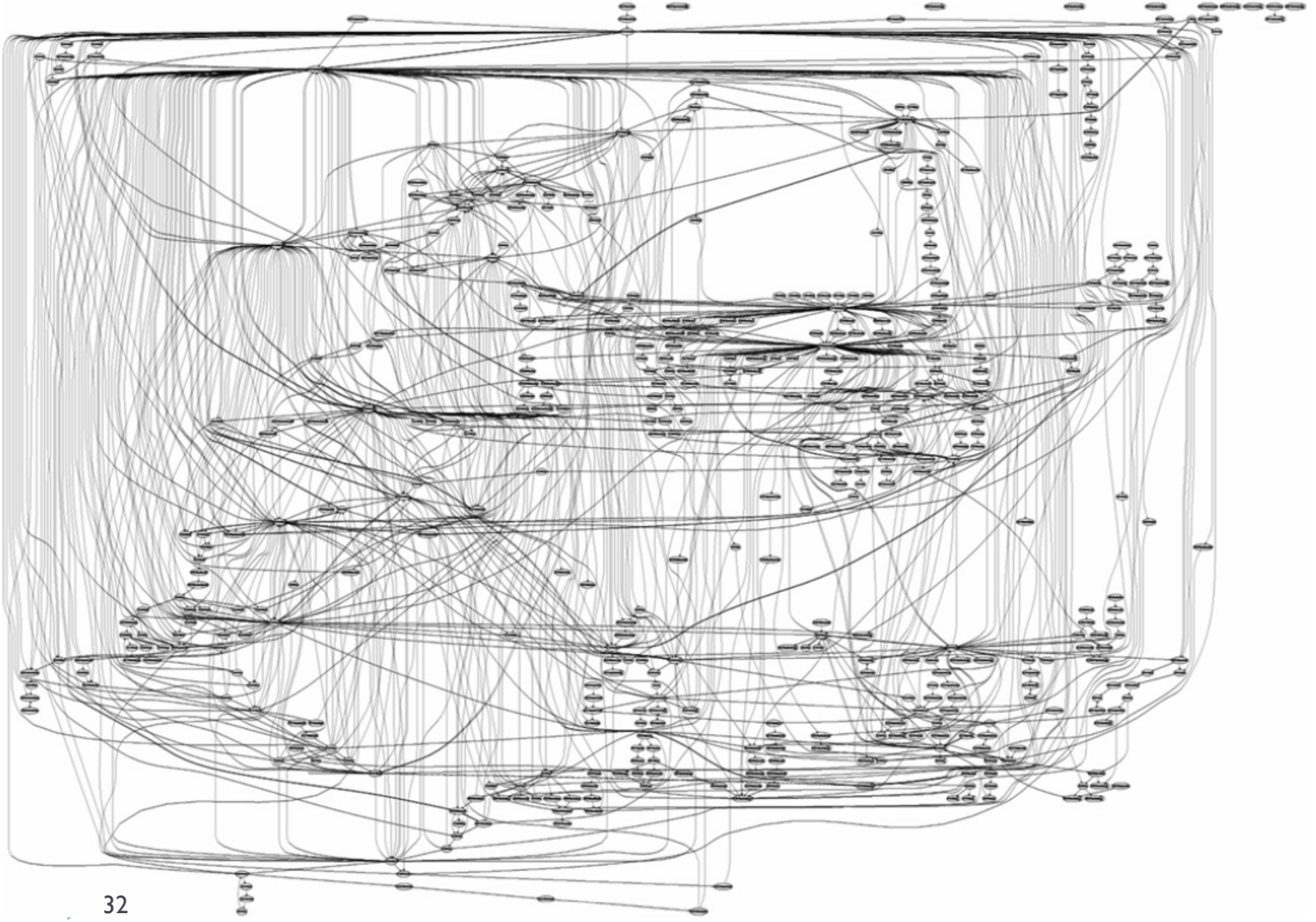
-O ein vorher gemessener Overhead wir gegeben, um die Genauigkeit der Statistik zu verbessern

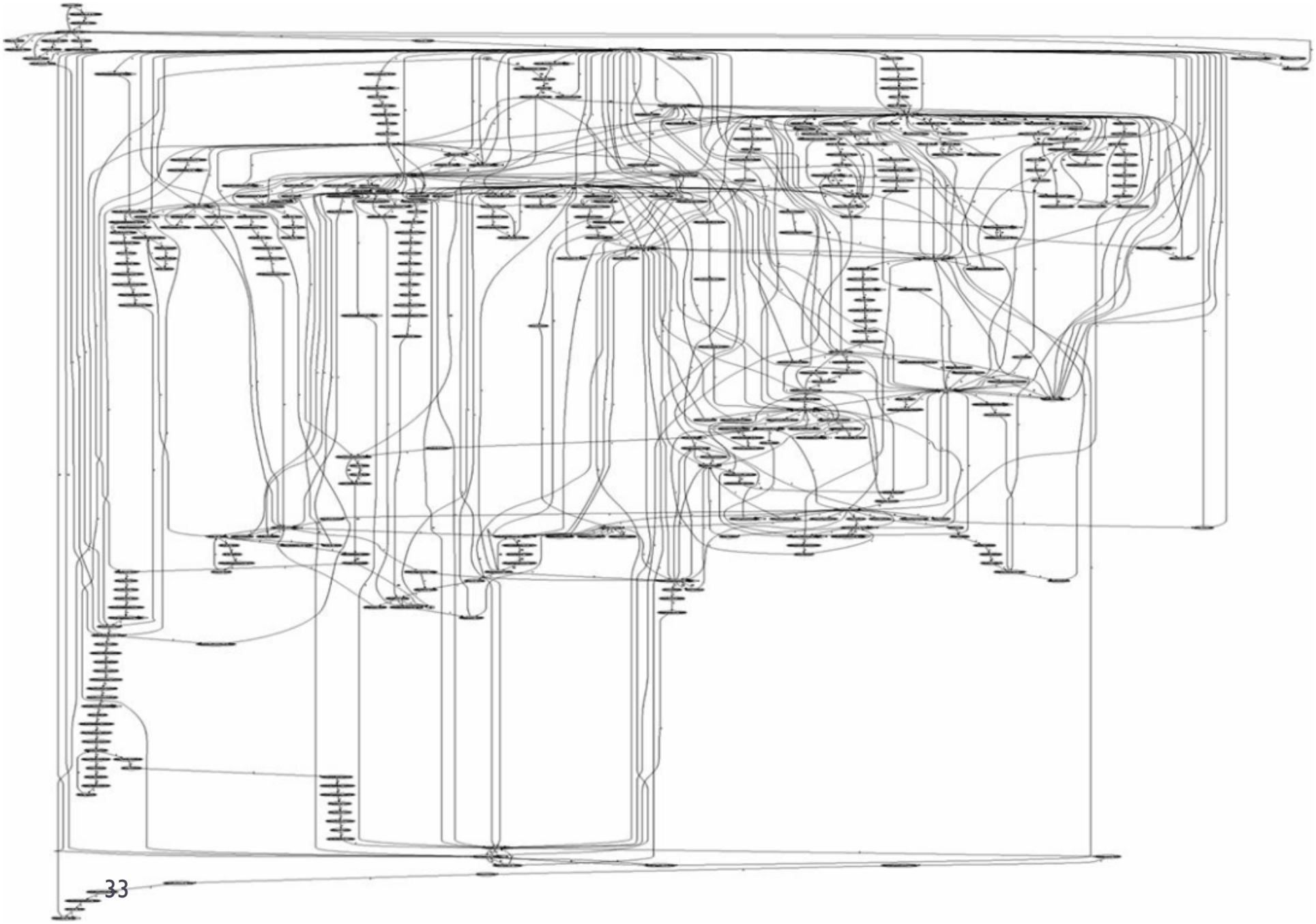
-u startet den Prozess mit der uid und gid des gegebenen Users

-E Systemvariablen für den Prozess mit geben

strace

Demo





Frage, Antworten, Diskussionen …

… und ansonsten, besten Dank für‘s zuhören!

Quellen

- In Reihenfolge des Auftretens:
  - http://research.intelego.net/heise/ix/1999/03/120/art.htm
  - http://de.wikipedia.org/wiki/Monolithischer_Kernel (wird mehrfach verwendet)
  - http://www.ibm.com/developerworks/linux/library/l-linux-kernel/
  - Die beiden letzten Graphen:
  - http://thermalnoise.wordpress.com/2006/05/09/syscall-graph-apache-vs-iis/