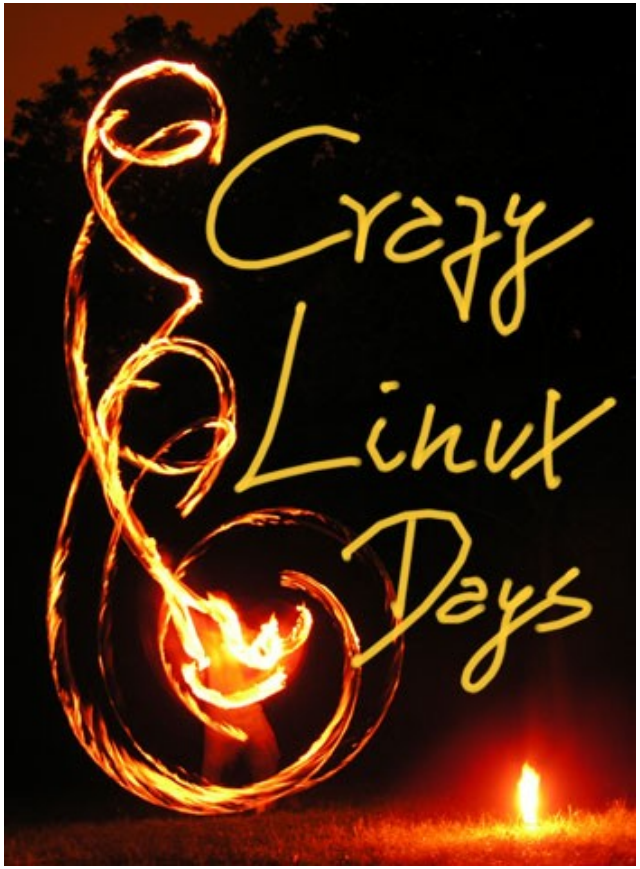
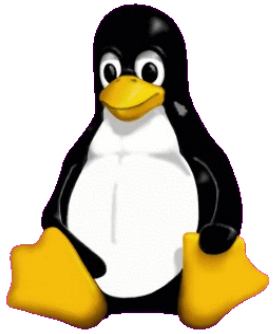
# Crazy Linux Days 2007

it-versity

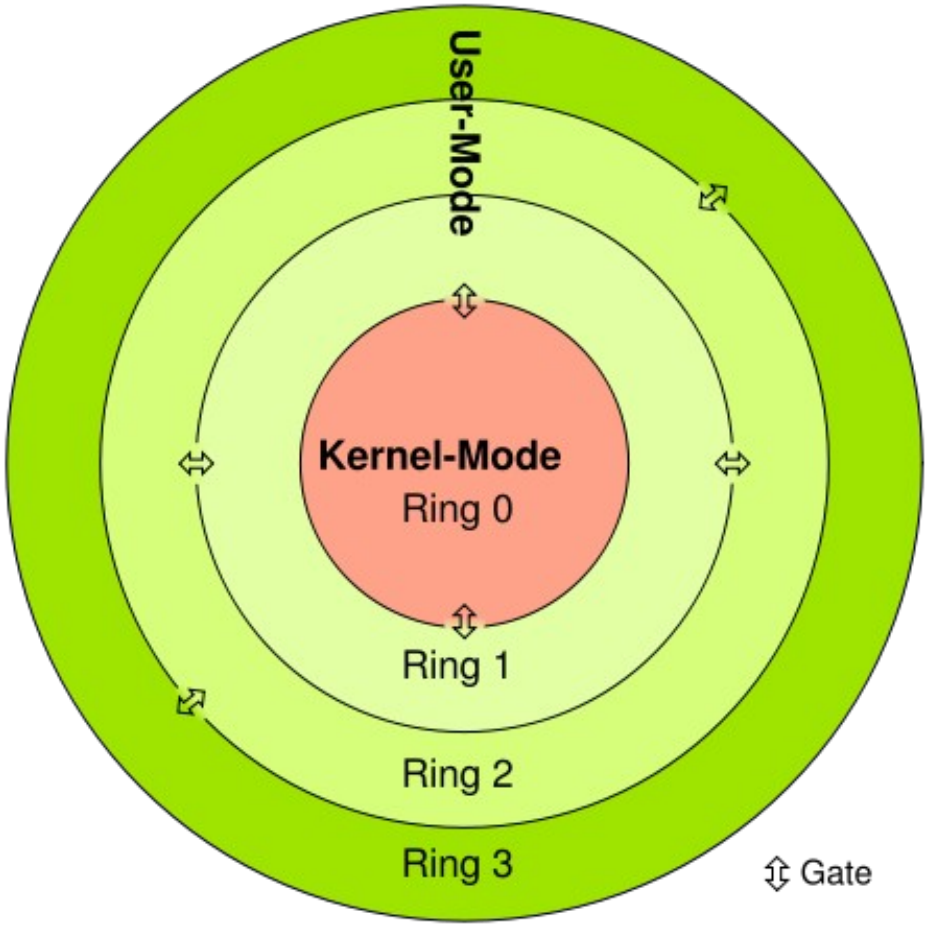



- **Debugging mit STRACE**
  - **Grundlagen**
  - **Systemaufrufe**
  - **Performance**



# Grundlagen

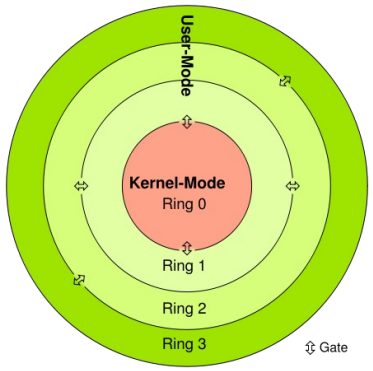
# Kernel Ring Modell




- Ring 1-3
  - User-Modus
  - unprivilegierter Modus
- Ring 0
  - Kernel Modus
  - privilegierter Modus

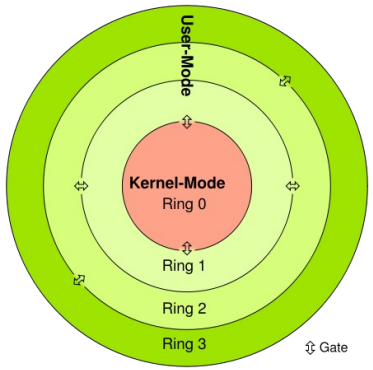
# privilegierter Modus

- privilegierter Modus
  - In diesem Modus steht dem Kernel der komplette CPU-Befehlssatz zur Verfügung
  - dieser Modus darf auf die komplette Hardware und den Speicher zugreifen

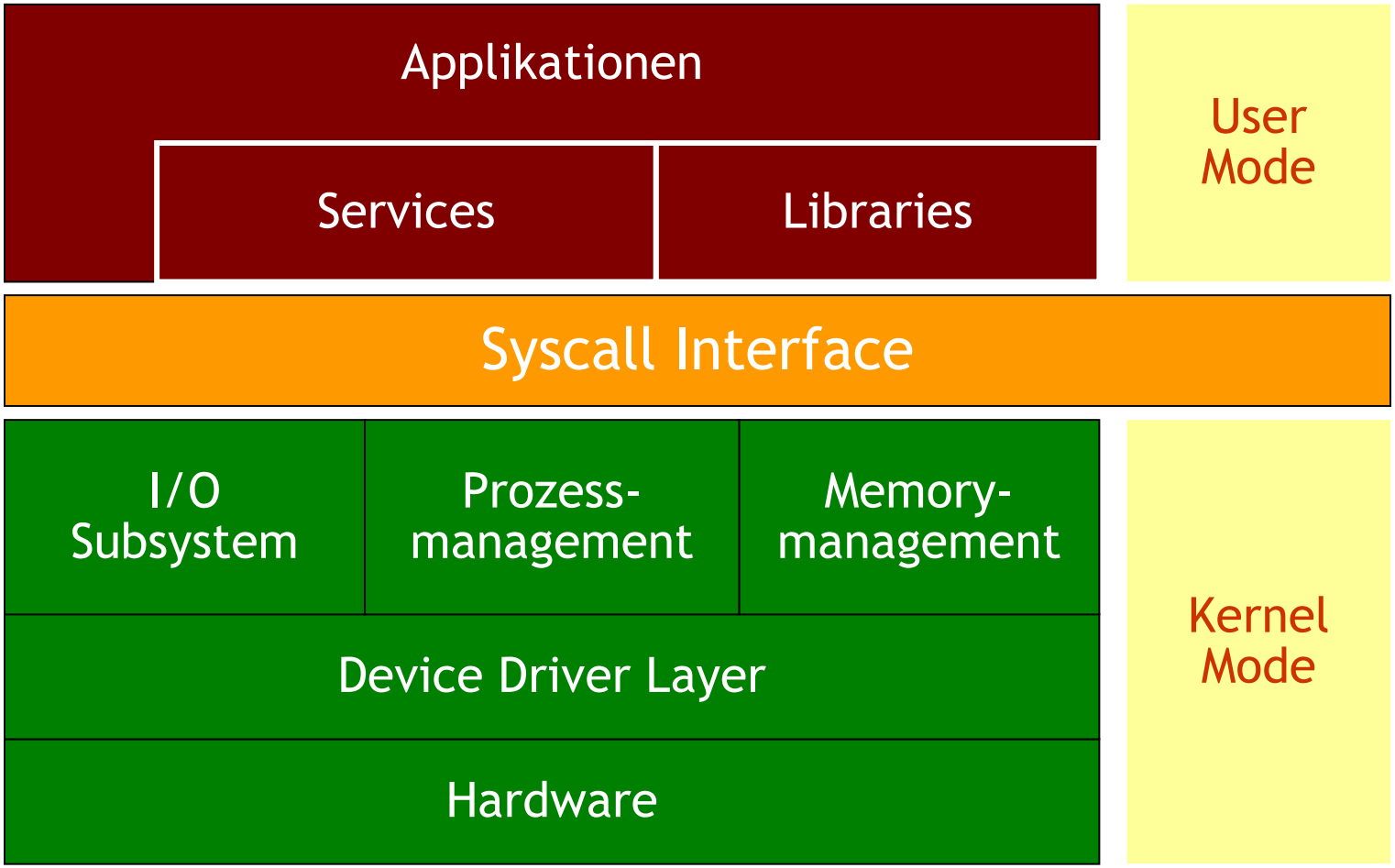


# unprivilegierter Modus

- unprivilegierter Modus
  - Benutzerprozesse laufen aus Sicherheit in diesem Modus, in dem weniger Befehle zur Verfügung stehen
  - Muss ein Benutzerprozess eine höhere Aufgabe erfüllen so sendet er dem Kernel einen Systemaufruf (Syscall)



# Syscall

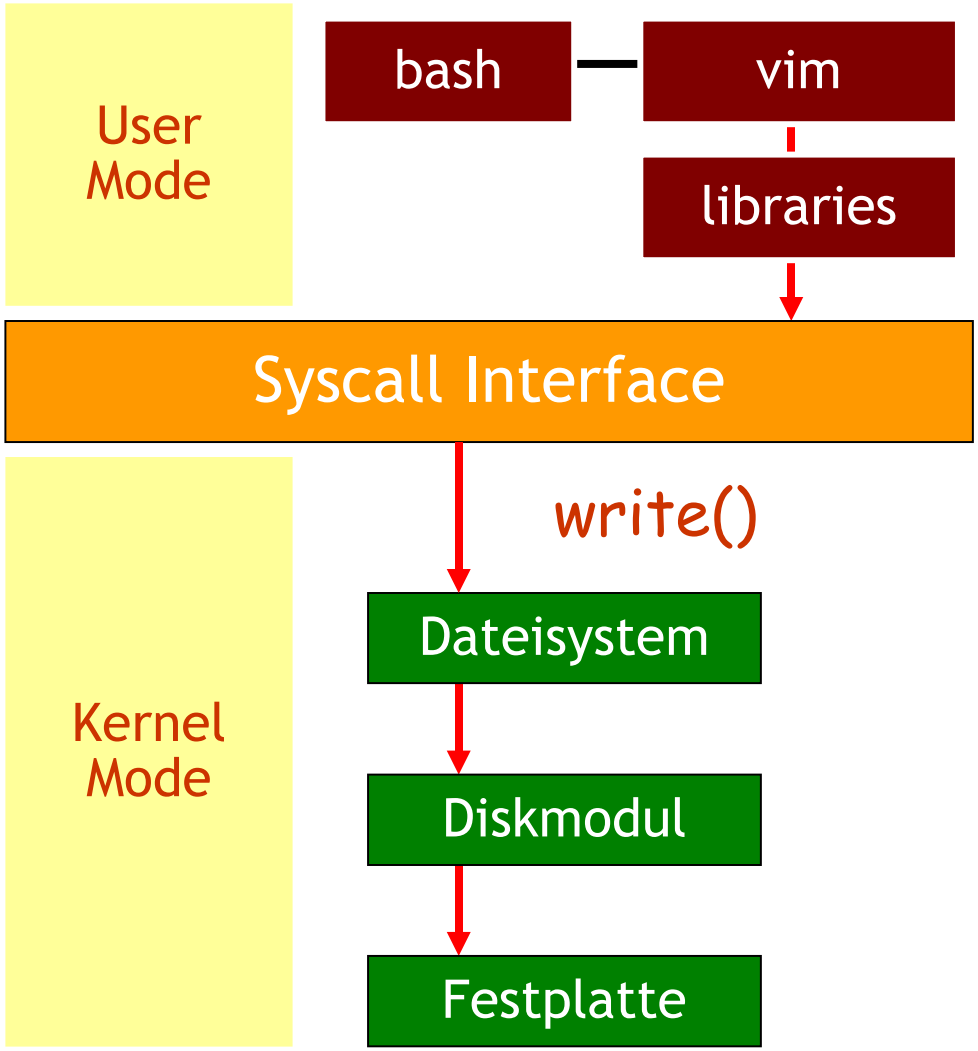


# Systemcall-Interface

it-versity

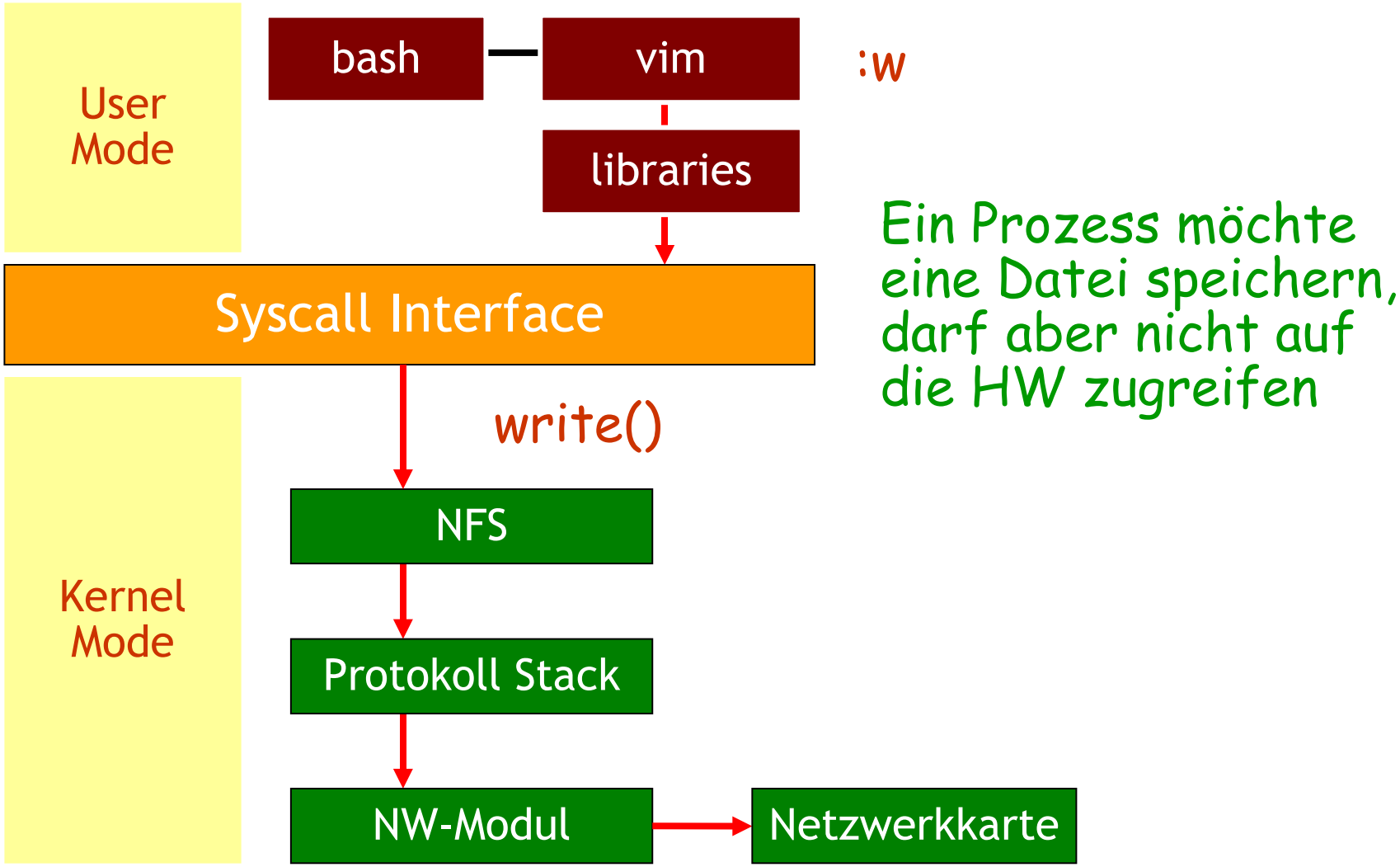

- Systemcall-Interface (Dienstzugangsschnittstelle)
  - Über die Systemcall-Schnittstelle lassen sich aus der Applikation heraus die Dienste des Betriebssystems nutzen.
  - Diese Schnittstelle ist absolut unabhängig von jeglicher Programmiersprache. In den seltensten Fällen greift eine Applikation direkt auf das Systemcall-Interface zu, sondern nutzt zu diesem Zweck Bibliotheksfunktionen.
  - So lautet beispielsweise der Systemcall für die Ausgabe von Daten in eine Datei oder in ein Gerät write()

# Beispiel 1 - lokal speichern




Ein Prozess möchte eine Datei speichern, darf aber nicht auf die HW zugreifen

# Beispiel 2 - remote speichern



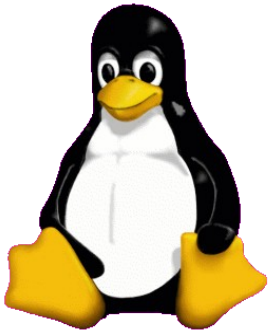

Ein Prozess möchte eine Datei speichern, darf aber nicht auf die HW zugreifen

- Was ist strace?
  - strace ist ein Programm und schaltet sich zwischen einen Benutzerprozeß und den Kernel und protokolliert alle Aktivitäten an dieser Schnittstelle.
  - Wenn der verfolgte Prozeß einen Systemaufruf macht, gibt strace den Namen, die Argumente des Aufrufs und den Rückgabewert dieses Systemaufrufs aus.
  - Wenn der verfolgte Prozeß ein Signal erhält, gibt strace den Namen des Signals aus.

# Arten von trace

- strace
  - protokolliert Systemaufrufe eines Prozesses
- ltrace
  - protokolliert Bibliotheksaufrufe und Systemaufrufe eines Prozesses

- Anwendungsgebiete von strace / ltrace
  - Debugging von Programmen
  - Debugging von laufenden Prozessen
  - Systemanalyse



# strace

# Ausgabe von strace

```
# Einsatz von strace
peter@asterix:~>  strace /bin/true
execve("/bin/true", ["/bin/true"], [/* 82 vars */]) = 0
brk(0)                                  = 0x804d000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=119279, ...}) = 0
mmap2(NULL, 119279, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7f23000
close(3)                                = 0
open("/lib/libc.so.6", O_RDONLY)        = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0@a\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1281488, ...}) = 0
mmap2(NULL, 4096, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7f22000
mmap2(NULL, 1254896, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb7def000
fadvise64(3, 0, 1254896, POSIX_FADV_WILLNEED) = 0
mmap2(0xb7f1c000, 12288, MAP_DENYWRITE, 3, 0x12c) = 0xb7f1c000
mmap2(0xb7f1f000, 9712, MAP_ANONYMOUS, -1, 0) = 0xb7f1f000
close(3)                                = 0
...
```

# Syntax von strace

it-versity

**strace optionen befehl argumente**

- protokolliert Systemaufrufe eines Prozesses

| OPTIONEN | |
|---|---|
| **-p #** | dockt an den Prozess Nr # an |
| **-o D** | lenkt die Ausgabe in Datei D um |
| **-f** | folgt auch Unterprozessen |
| **-e A,B,...** | zeigt nur die Systemaufrufe A,B,... |
| **-c** | Statistik |
| **-v** | zeigt die Argument ungekürzt an (verbose) |

# execve

```
# Einsatz von strace
peter@asterix:~>  strace /bin/true
execve("/bin/true", ["/bin/true"], [/* 85 vars */]) = 0
....
```

- execve
  - Dateinamen des Programmes /bin/true
  - Liste der Argumente [/bin/true] wobei das erste Argument der Name des Programmes oder Prozesses ist
  - Liste der Umgebungsvariablen (/proc/PID/environ) wird durch /* # vars */ angedeutet
  - Rückgabewert  = 0

# Rückgabewerte

```
# Einsatz von strace
peter@asterix:~>  strace /bin/true
execve("/bin/true", ["/bin/true"], [/* 85 vars */]) = 0
....
_exit(0) = ?
```

- _exit(0) = ?  oder  exit_group(0) = ?
  - Letzte Zeile enthält keinen Rückgabewert mehr
  - (0)  ... ist der Rückgabewert der an den Vaterprozess gemeldet wird
  - kann mit echo $? abgefragt werden

# Übung

- Vergleichen von Programmen

strace -o /tmp/true.trace /bin/true

strace -o /tmp/false.trace /bin/false

diff /tmp/{true,false}.trace

Sind die exit Zeilen gleich?

# Generelles

- Generelles zu strace
  - jede Zeile protokolliert einen Systemaufruf
  - jeder Aufruf gibt einen Rückgabewert zurück
  - ein negativer Rückgabewert ist ein Fehler
  - es gibt ~263 verschiedene Systemaufrufe
  - Dokumentation zu Systemaufruf open

    man 2 open

# Filtern von Systemaufrufen

it-versity

- Filtern von Systemaufrufen

  strace -e execve,_exit /bin/true

  strace -e execve,exit_group /bin/true

```
# Einsatz von strace mittels Filter
peter@asterix:~>  strace -e execve,exit_group  /bin/true
execve("/bin/true", ["/bin/true"], [/* 85 vars */]) = 0
exit_group(0) = ?
```

# Aufgabe

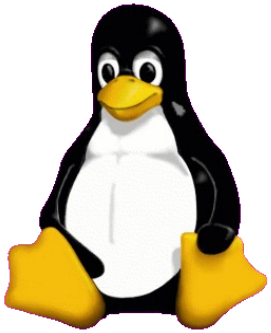
- /tmp/true2.trace  erstellen
  - auszuführendes Programm: /bin/true
  - Filter: execve, exit_group
- /tmp/false2.trace  erstellen
  - auszuführendes Programm: /bin/false
  - Filter: execve, exit_group
- Beide traces mit diff vergleichen

# Arten von Systemaufrufen

- Dateizugriffe
  - open, read, write, close
- Verzeichnisse- & Dateiattribute
  - chdir, mkdir, rename, unlink, ...
  - chmod, chown, stat, fstat, lstat, ...
- Netzwerk
  - socket, bind, listen, accept, connect

# Arten von Systemaufrufen

- UIDs & GIDs
  - setguid, getuid, setgid, getid
- Prozessverwaltung
  - select, fork, wait, execve
- Diverse
  - mmap, munmap, mprotect, brk, uname,...



# Systemaufrufe im Detail

## - open, close -

# Systemaufrufe: Dateizugriffe

```
# Einsatz von strace
open("/etc/ld.so.cache", O_RDONLY)      = 3
```

- open
  - erhält als Argument einen Dateinamen und Zugriffsmodus und liefert als Rückgabewert eine positive Zahl den Dateideskriptor
  - Eine negative Zahl als Rückgabewert ist ein Fehler
  - Zeigt welche Konfigurationsdateien gelesen werden
  - Zeigt in welche (Log-) Dateien geschrieben wird

# OPEN

```
# Einsatz von strace
peter@asterix:~>  strace -e execve,open /bin/true
execve("/bin/true", ["/bin/true"], [/* 85 vars */]) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/lib/libc.so.6", O_RDONLY)        = 3
...
```

| OPEN FLAGS | |
|---|---|
| **O_RDONLY** | Datei nur zum LESEN öffnen |
| **O_WRONLY** | Datei nur zum SCHREIBEN öffnen |
| **O_RDWR** | Datei zum LESEN & SCHREIBEN öffnen |
| können noch zusätzlich mit anderen flags kombiniert werden | |

# Fehlerbehebung mit open

- Fehlerbehebung mit open
  - Manche Programme klappern eine ganze Latte von möglichen Orten nach Konfigurationsdateien ab
- Aufgabe
  - welche Konfigurationsdateien liest ssh localhost?
  - welche hat eine größere Priorität?

# Fehlermeldungen bei open

it-versity

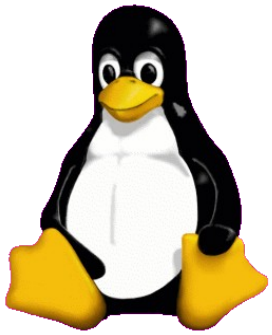
```
# Beispiel für eine nicht gefundene Datei
open("/root/.cups/client.conf", O_RDONLY|O_LARGEFILE) = -1 ENOENT \
      (No such file or directory)
```

| FEHLERMELDUNGEN BEI OPEN | |
|---|---|
| **EEXIST** | pathname existiert bereits |
| **EISDIR** | pathname bezieht sich auf einen Verzeichnisnamen |
| **EACCESS** | Der Zugriff wurde verweigert |
| **ENOENT** | Der angegebene Verzeichnispfad existiert nicht |
| **EROFS** | Schreibfehler da read-only Dateisystem vorliegt |
| mehr FLAGS: man 2 open | |

# Systemaufruf: close

```
# Einsatz von strace
peter@asterix:~>  strace -e open,read,close   ssh localhost
open("/etc/ssh/ssh_config", O_RDONLY|O_LARGEFILE) = 3
read(3, "#\t$OpenBSD: ssh_config,v 1.22 20"..., 4096) = 2643
read(3, "", 4096)                       = 0
close(3) 0
```

- Systemaufruf: close(#)
  - Ist das Gegenstück zu open
  - Unbedeutend für die Fehlersuche
  - Danach ist der Deskriptor X wieder frei und kann bei einem anderen open verwendet werden



# Systemaufrufe im Detail

## - read, write -

# Systemaufruf: read, write

- Systemaufruf: read, write
  - dienen zum lesen und schreiben in durch open geöffneten Dateien
  - als Dateiname wird der Dateidiskriptor von open verwendet
  - zeigt welcher Datenstrom wirklich gelesen oder geschrieben wird

# open & read

```
# Beispiel für open & read
open("/root/.ssh/known_hosts", O_RDONLY|O_LARGEFILE) = 4
read(4, "mail.triples.at,81.223.158.165 s"..., 4096) = 925
read(4, "", 4096)
```

open (dateiname, OPEN_FLAGS) = Dateideskriptor

read (Dateideskriptor, "INHALT..., Bytes) = Bytes

# Systemaufruf: read, write

it-versity

```
read (3, "xx 32bytes xx", 4096) = 1823
```

- Systemaufruf: read, write
  - strace zeigt als Debuginfo nur die ersten 32Byte an die gelesen oder geschrieben werden
  - kann mit der Option -s 500 z.B. auf z.b 500 Zeichen/Bytes erweitert werden

# Übung

- Normale read Operation

```
strace -o pass.trace -e open,read  cat /etc/passwd
```

- read Operation mit 1000 Zeichen

```
strace -s 1000 -o pass2.trace -e open,read   \
cat /etc/passwd
```

# read & bytes

```
# Beispiel für open & read
peter@asterix:~>  strace -o pass.trace -e open,read  cat /etc/passwd
 open("/etc/passwd", O_RDONLY|O_LARGEFILE) = 3
 read(3, "at:x:25:25:Batch jobs daemon:/va"..., 4096) = 1823
 read(3, "", 4096)                       = 0
```

read (3, "xx 32bytes xx", 4096) = 1823

Lese-Blockgröße in Bytes

wirklich gelesene Bytes

```
# Wie groß ist die Datei passwd?
peter@asterix:~>  ls -l /etc/passwd
-rw-r--r-- 1 root root 1823 17. Okt 14:56 /etc/passwd
```

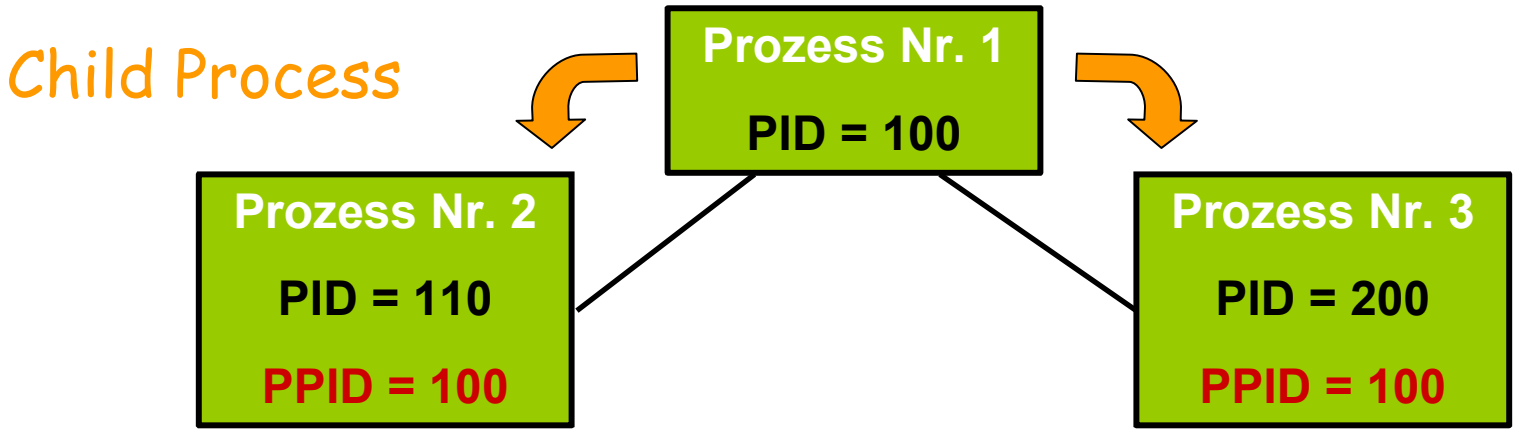
# Übung

- Wo ist der Unterschied zwischen:
  - dd if=/dev/zero of=/dev/null bs=300000 count=1
  - dd if=/dev/zero of=/dev/null bs=1 count=300000
- Testen mit
  - strace -e open,read,write

# Dämon überwachen

- Welche Konfigurationsdateien liest samba?
  - rcsmb stop
  - strace -o samba.read -e open,read rcsmb start
- In welche Dateien  schreibt samba?
  - rcsmb stop
  - strace -o samba.write -e open,write rcsmb start

# Dämon überwachen 2




- Folgen von Kinderprozessen
  - rcsmb stop
  - strace -f -o samba.read -e open,read rcsmb start &
  - strace -f -o samba.write -e open,write rcsmb start &

strace muss mit kill aus einer anderen Konsole beendet werden

oder mit & als Hintergrundprozess gestartet werden

# open & write

```
# Beispiel für open & write
open("/var/log/samba/log.smbd", O_WRONLY|O_CREAT| \
      O_APPEND|O_LARGEFILE, 0644) = 4
write(4, "[2007/11/19 10:58:26, 0] smbd/se"..., 49) = 49
write(4, "  smbd version 3.0.26a-3-1478-SU"..., 51) = 51
write(4, "  Copyright Andrew Tridgell and "..., 57) = 57
```

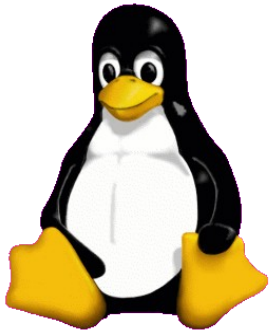
open (dateiname, OPEN_FLAGS) = Dateideskriptor

write (Dateideskriptor, "INHALT..., Bytes) = Bytes

# Öffnen einer LOG Datei

```
# Beispiel für das öffnen einer LOG Datei
open("/var/log/samba/log.smbd", O_WRONLY|O_CREAT| \
     O_APPEND|O_LARGEFILE, 0644) = 4
```

| ERWEITERTE OPEN FLAGS | |
|---|---|
| O_RDONLY, O_WRONLY, O_RDWR können erweitert werden durch | |
| **O_CREAT** | Wenn die Datei nicht existiert wird sie erstellt |
| **O_TRUNC** | Wenn die Datei existiert wird sie überschrieben |
| **O_APPEND** | Öffnen der Datei im Anhängemodus |
| **O_SYNC** | synchron I/O Modi, der Prozess wird solange angehalten bis die Datei physikalisch geschrieben ist |
| **0664** | Rechte die beim erstellen gesetzt werden |
| mehr FLAGS: man 2 open | |



# Systemaufrufe

## - chdir, mkdir, rmdir, -

## - rename, unlink -

# Verzeichnisse & Dateien

- chdir
  - wechselt das aktuelle Arbeitsverzeichnis
- mkdir, rmdir
  - Erstellt bzw. löscht ein leeres Verzeichnis
- unlink
  - löscht eine Datei (entfernt Inode Eintrag)
- rename
  - Umbennen oder verschieben innerhalb des Dateisystems

# Datei

- Verschieben einer Datei
  - echo "Kleine Datei" > datei.alt
  - strace -e rename  mv datei.alt  datei.neu

    open,close,read,write,rename,unlink

- Was passiert wenn:
  - die Datei auf der selben Partition verschoben wird
  - auf eine andere Partition verschoben wird

# Verschieben

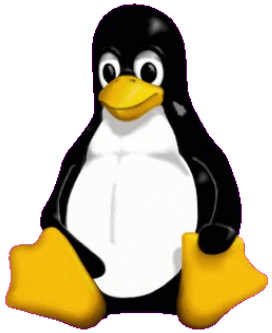
- Verschieben auf eine andere Partition
  - rename schlägt mit Fehler EXDEV fehl
  - Original wird geöffnet
  - Ziel wird geöffnet
  - Original wird gelesen und in Ziel geschrieben
  - Dateien werden geschlossen
  - Original wird entfernt (unlink)

# ÜBUNG

- Kopieren von Dateien
  - wo liegt der Unterschied zwischen dem Verschieben und dem kopieren einer Datei auf eine andere Partition?

# Dateiattribute

- chmod, chown, stat, stat64
  - dienen zum abfragen und ändern von Dateiattributen. (64 ist 64-bit Version)
- fchmod, fchown, fstat und fstat64
  - wie oben jedoch wird anstelle eines Dateinamens der Dateidiskriptor von open verwendet
- lchown, lstat, lstat64
  - verhalten sich anders bei symbolischen Links



# Performance

# Strace Statistik

```
# Ausgabe einer Statistik
peter@asterix:~>  strace -c   ssh localhost
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 94.76    0.003999         190        21           fadvise64
  5.24    0.000221           5        49        14 open
  0.00    0.000000           0       107           read
  0.00    0.000000           0        42           write
  0.00    0.000000           0        48           close
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         3           time
  0.00    0.000000           0         1         1 access
  ...
```

# Statistik

```
peter@asterix:~> strace -c  ssh localhost
% time     seconds  usecs/call     calls    errors     syscall
------ ----------- ----------- --------- --------- -------  ---------
 94.76    0.003999         190        21                   fadvise64
  5.24    0.000221           5        49        14         open
```

| OPTIONEN | |
|---|---|
| **%time** | Gesamtzeit verbraucht durch diesen Systemcall |
| **seconds** | Gesamtzeit in Sekunden |
| **usecs/call** | Microsekunden pro Systemcall |
| **calls** | Gesamtzahl diesen Typs |
| **errors** | Anzahl der Fehler bei diesem Systemcall |

# Zeitmessung

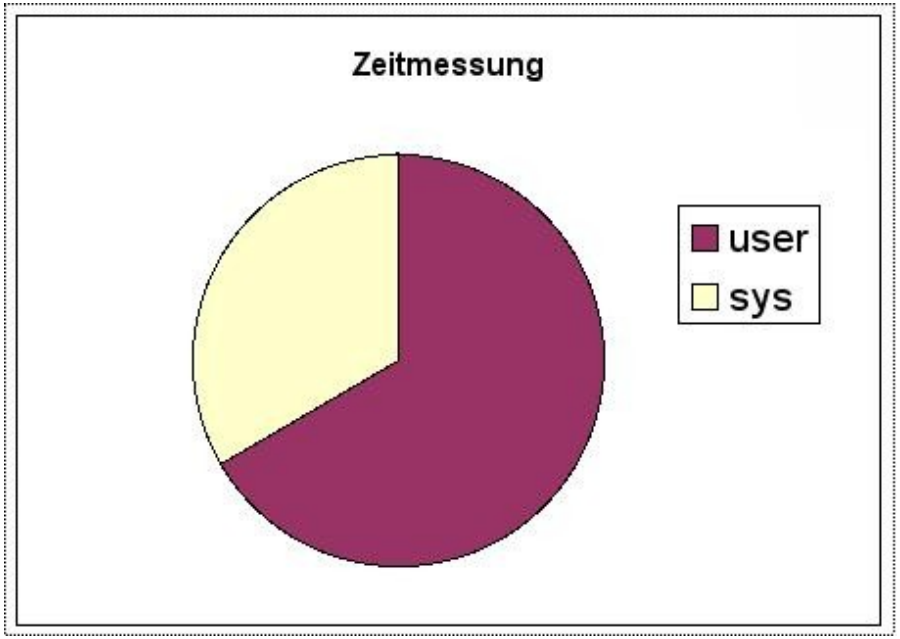
it-versity

## time Befehl Argumente

- Der Befehl "time"
  - misst wie lange ein Befehl für die Ausführung braucht
  - liefert 3 Rückgabewerte

real x Millisekunden
user x Millisekunden
sys x Millisekunden

# Rückgabewerte von time




| ZEIT | BEDEUTUNG |
|---|---|
| real | zeigt die real verbrauchte Zeit (Gesamtzeit) |
| user | vom Prozess verbrauchte Zeit im Usermodus |
| sys | vom Prozess verbrauchte Zeit im Kernelmodus |

# Übung

```
# Zeitmessung eines find-Befehles
peter@tux~> time  find /etc -name "*.conf"
/etc/reader.conf.d/reader.conf
/etc/powerd.conf
/etc/ntp.conf
/etc/wvdial.conf
...
real    0m3.798s
user    0m0.076s
sys     0m0.272s
```

# Ergebnis

- Interpretation der Rückgabewerte
  - user + sys = vom Prozess verbrauchte Zeit
  - user + sys = entspricht nicht der real Zeit!

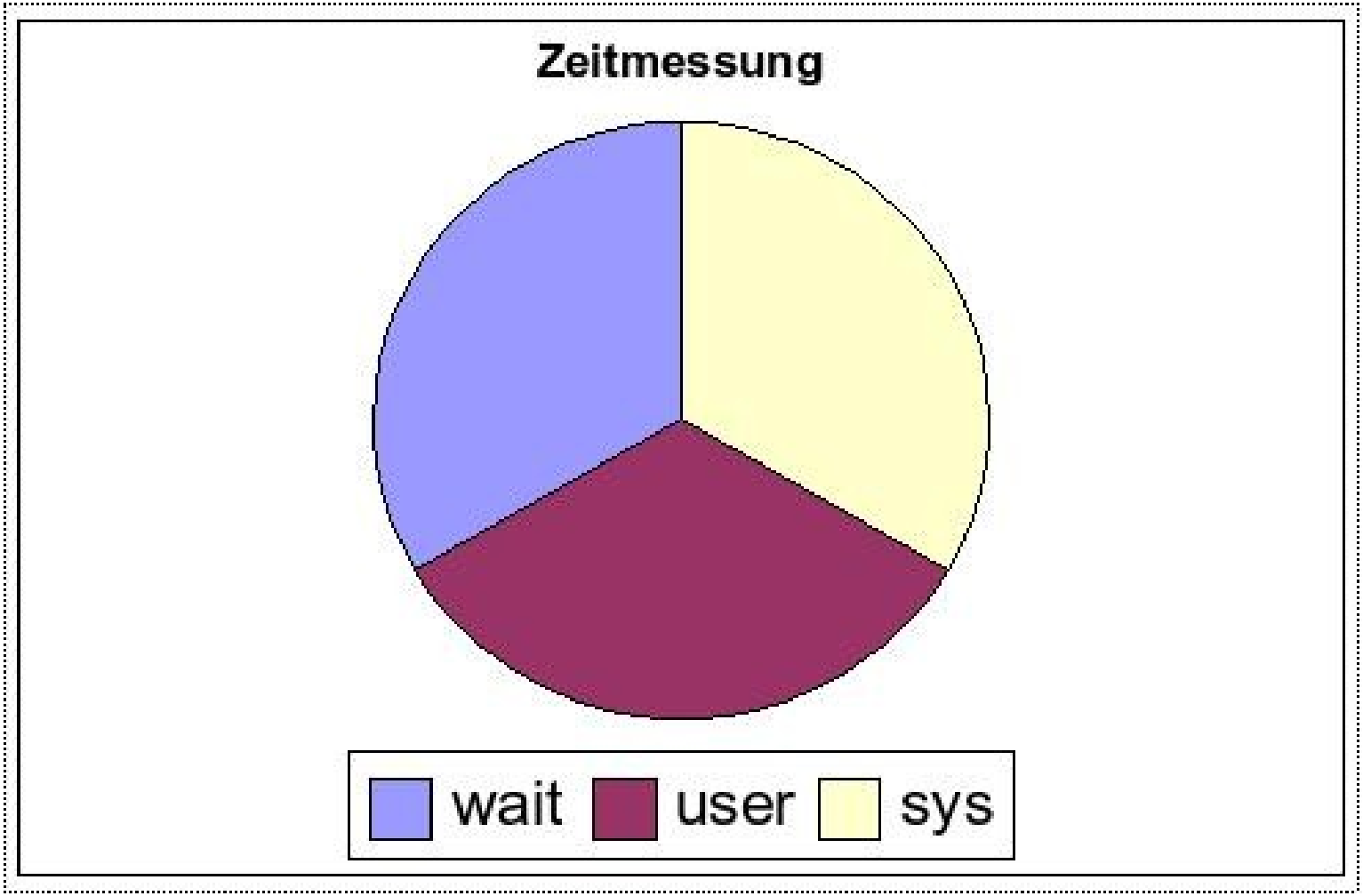
```
real    0m3.798s
user    0m0.076s
sys     0m0.272s
```

Wo ist die restliche Zeit verschwunden?

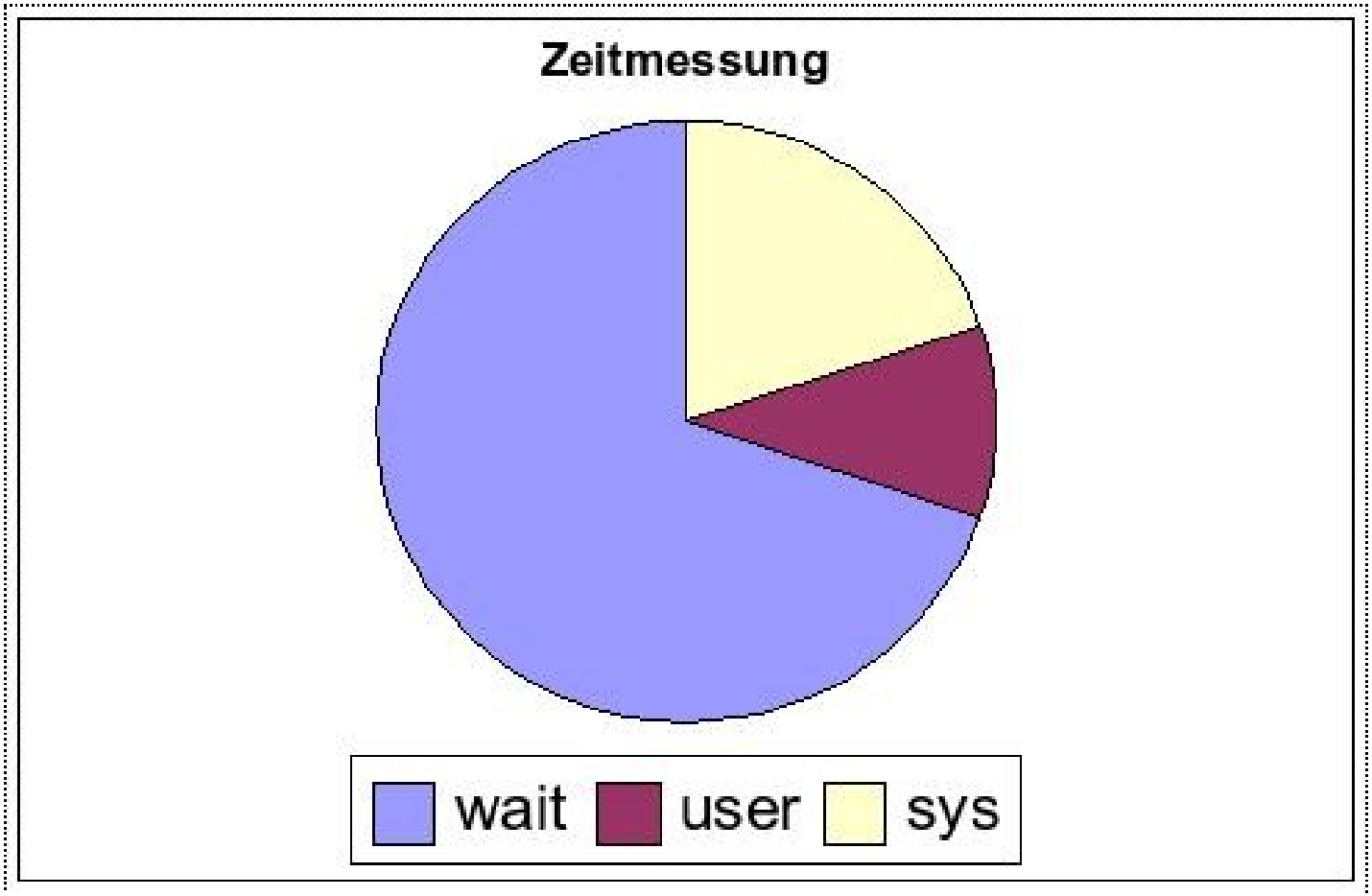
# Die fehlende Zeit

- Der Zustand "wait"
  - ein Systemaufruf versetzt einen Prozess sehr oft in den wait-Zustand
  - in dieser Zeit verbraucht der Prozess keine CPU-Zeit
  - Speicher wird in dieser Zeit nicht freigeben
- Folgen
  - belastet nicht die CPU
  - belastet die Nerven des Benutzers

# user + sys + wait = real



# hoher wait-Anteil



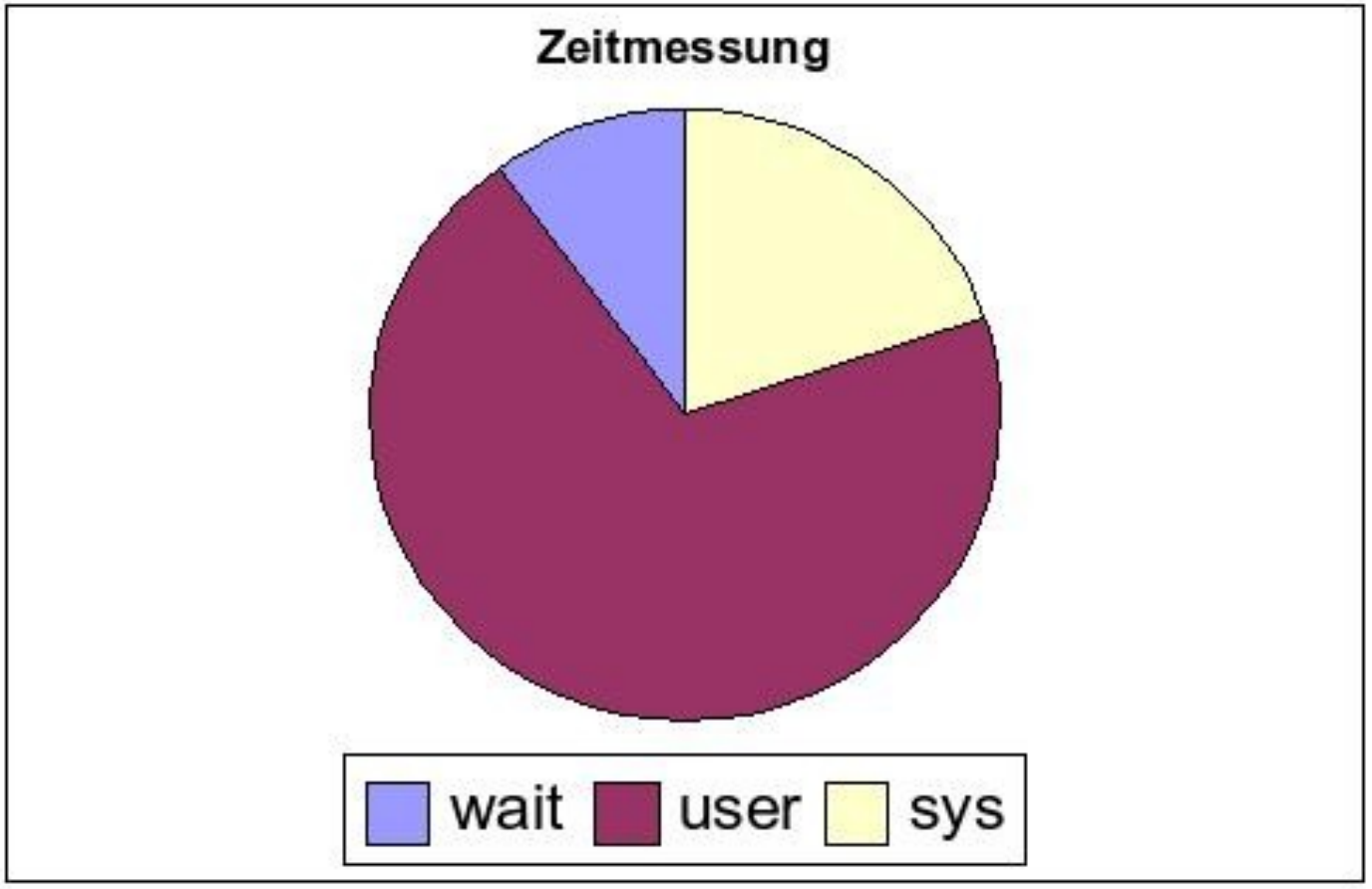
# hoher wait-Anteil

- Ursache für den Zustand "wait"
  - Prozess wartet auf ein Ereignis
    - auf die Benutzereingabe
    - auf ein Gerät
    - ein Netzwerkpaket
    - ....
  - Wartet darauf CPU-Zeit zu bekommen
    - abhängig von Prozesspriorität, nice-Wert, Anzahl
    - abhängig von freien Speicher und CPU-Auslastung

## Ist ein hoher wait-Anteil schlecht?

- NEIN
  - der Konqueror wird gestartet und er wird nicht benutzt. Belastet nicht meine CPU
- JA
  - ein Backupjob wird gestartet und er muss andauernd auf die Daten warten. lange Backupzeit

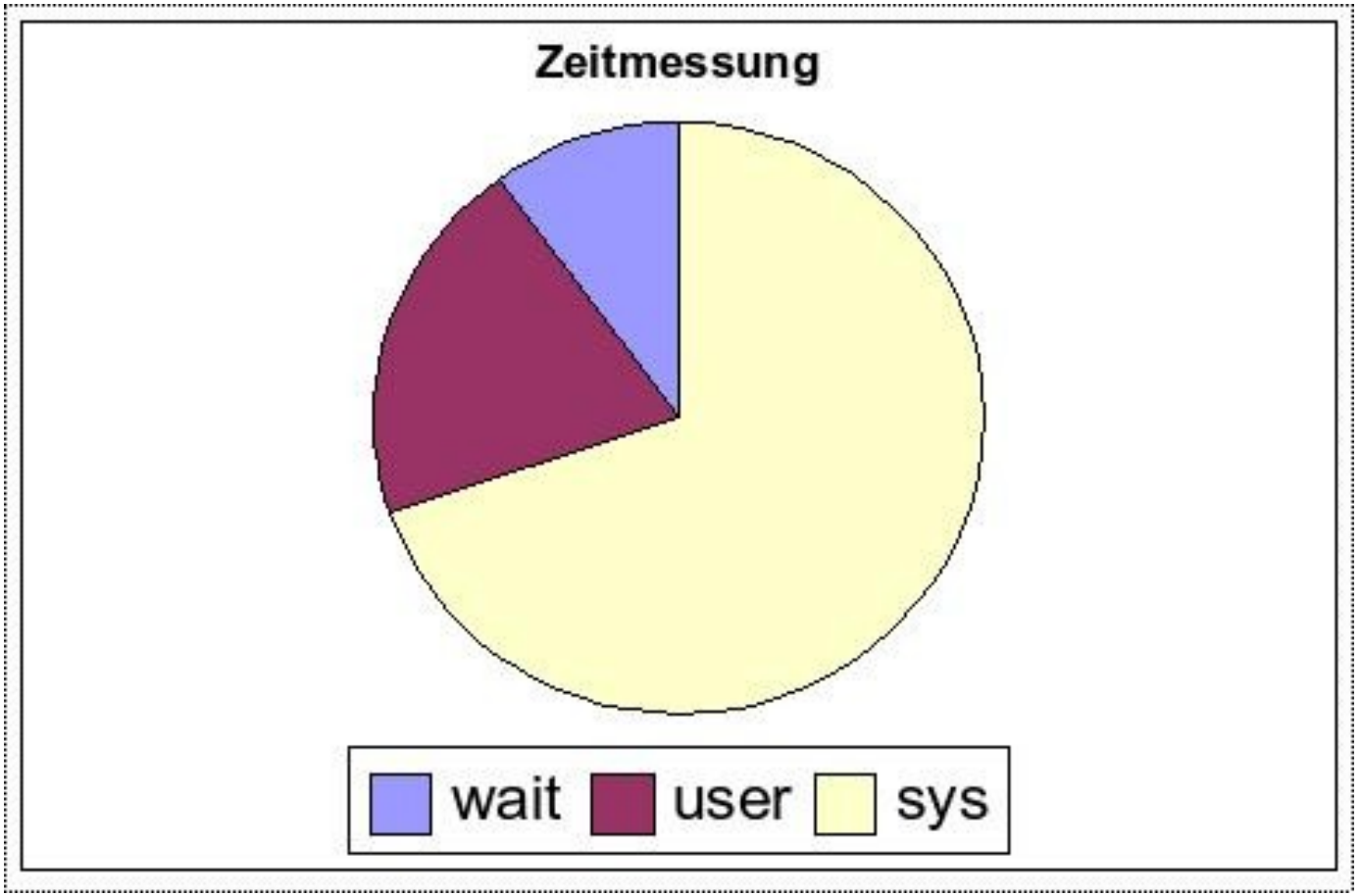
# hoher user-Anteil



# hoher user-Anteil

- Mögliche Ursachen für hohen user-Anteil
  - aufwändige Berechnungen
    - verschlüsseln von Daten
    - komplizierte Algorithmen
    - ...
  - schlechte Programmierung
    - primitiver Sortieralgoritmus
    - schlechte Interpretersprache (C ist schneller wie bash)
    - ...

# hoher sys-Anteil



# hoher sys-Anteil

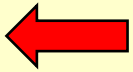
- Mögliche Ursachen für hohen sys-Anteil
  - Ein-/Ausgabeoperationen mit schlechtem Übertragunsmodus. z.b ohne DMA
  - komplexe Dateisystemoperationen
    - verschlüsseln des Dateisystems
    - Dateisystemcheck
    - ...
  - viele Zugriffe auf die Hardware
  - unnötig viele Systemaufrufe

# Übung

- Übung
  - Kopieren von 300kB von /dev/zero nach /dev/null
- Beide Varianten mit time testen
  - dd if=/dev/zero of=/dev/null bs=300000 count=1
  - dd if=/dev/zero of=/dev/null bs=1 count=300000

# Ergebnis

```
# Kopieren von 300kB
peter@asterix:~>  time dd if=/dev/zero of=/dev/null bs=300000 count=1
1+0 Datensätze ein
1+0 Datensätze aus
300000 Bytes (300 kB) kopiert, 5,8945e-05 s, 5,1 GB/s

real    0m0.004s
user    0m0.000s
sys     0m0.004s
```



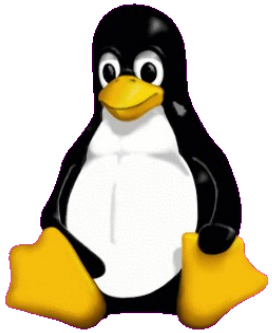
```
# Kopieren von 300kB
peter@asterix:~> time dd if=/dev/zero of=/dev/null bs=1 count=300000
300000+0 Datensätze ein
300000+0 Datensätze aus
300000 Bytes (300 kB) kopiert, 0,231105 s, 1,3 MB/s

real    0m0.234s
user    0m0.068s
sys     0m0.164s
```

# time & strace

- time
  - time zeigt nur eine kurze Zusammenfassung
- strace
  - strace -c zeigt eine detailierte Aufstellung
  - strace zeigt genaue Informationen zu jedem ausgeführten Systemcall an
  - perfekt für das Troubleshooting und das verbesseren der Performance

# Strace Statistik

```
# Ausgabe einer Statistik
peter@asterix:~>  strace -c   ssh localhost
% time     seconds  usecs/call     calls    errors     syscall
------ ----------- ----------- --------- --------- -------   ------------
 94.76    0.003999         190        21              fadvise64
  5.24    0.000221           5        49        14    open
  0.00    0.000000           0       107              read
  0.00    0.000000           0        42              write
  0.00    0.000000           0        48              close
  0.00    0.000000           0         1              execve
  0.00    0.000000           0         3              time
  0.00    0.000000           0         1         1    access
  ...
```



# ltrace

# Syntax von strace

it-versity

ltrace optionen befehl argumente

- protokolliert Bibliotheksaufrufe und Systemaufrufe eines Prozesses

| OPTIONEN | |
|---|---|
| -p # | dockt an den Prozess Nr # an |
| -o D | lenkt die Ausgabe in Datei D um |
| -f | folgt auch Unterprozessen |
| -e A,B,... | zeigt nur die Systemaufrufe A,B,... |
| -S | zeigt zusätzlich Systemaufrufe |
| -n X | rückt pro Aufrufebene um X Spalten ein |

# ltrace

```
# Ausgabe einer Statistik
peter@asterix:~> ltrace -c  find /usr/share/doc
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 86.27    1.071814          30     35327           write
 10.15    0.126092          38      3297           getdents64
  2.33    0.028931           3     10208           lstat64
  0.55    0.006861           2      3122         1 chdir
  0.39    0.004890           3      1567         2 open
[...]
  0.00    0.000003           3         1           uname
  0.00    0.000001           1         1           time
------ ----------- ----------- --------- --------- ----------------
100.00    1.242403                 58269         3 total  ...
```

# it-versity

## Wolfgang Ziegler und Partner Ges.m.b.H.

Schottenfeldgasse 69

A – 1070 Vienna

Phone +43 (1) 5228222

servicecenter@it-versity.com