$HELL WE CONTINUE?

Für mehr Shell im Alltag.

VORAUSSETZUNGEN:

Mit `cd`, `cp`, `mkdir`, `mv`, `rm`, `rmdir` umgehen können.

WAS ERWARTET DICH?

NICHT-ZIELE FÜR HEUTE

- Nicht bis ins letzte Detail gehen
- Keine Text-GUIs für E-Mail, Browser, … vorstellen
- Keine Behandlung von Bash-Arrays
- Kaum "Parameter Expansion" à la `${i/alt/neu}`
- Weder POSIX-Shell noch zsh; Fokus bleibt Bash

ZIELE FÜR HEUTE

- Bausteine und Konzepte vorstellen:
  - Path Resolution
  - Signale, Jobs, Expansion
  - Datenfluss / Piping, etwas Text-Processing
- Möglichkeiten und Fallen aufzeigen
- Praktisches für den Alltag in Bash, Skripting eher als Nebenprodukt

ZUM ABLAUF

- Kurze Pause nach 60 Minuten
- *Verständnisfragen* bitte gleich!

EINE BITTE:

BITTE KEINE *SOFTWARE* IN BASH SCHREIBEN.

- Langsam
- Schlechte Fehlerbehandlung
- Viel Ärger mit *Leerzeichen*
- Fließkomma-Zahlen (von Bash selbst) nicht unterstützt
- …

Man *kann* den nächsten Webserver in Bash schreiben, *sollte* aber nicht unbedingt.

TERMINAL "REPARIEREN"

RESET: TERMINAL ZURÜCKSETZEN

Terminal *kaputt machen* zum Beispiel mit:

```
# python -c 'import curses; curses.initscr()'
```

Zurücksetzen / *reparieren*:

```
<Ctrl+C>
# reset
```

AUFLÖSUNG VON PFADEN

WIE UND WARUM FINDET BASH "CP" UNTER "/BIN/CP"?

VERANTWORLICH IST EINE BESONDERE VARIABLE ${PATH}:

```
# echo "${PATH}"
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/u
```

AUFLÖSUNG VON PFADEN

```
# echo "${PATH}"
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/u
                           ^^^^

# which cp
/bin/cp


# type -p cp
/bin/cp
```

EIGENE PFADE MIT AUFNEHMEN

```
export PATH="${PATH}:${HOME}/bin"
```

(Permanent via `~/.bashrc`)

AUFLÖSUNG VON PFADEN

```
# echo -e '#! /bin/bash\necho Hello' > hello.sh
# chmod +x hello.sh

# ./hello.sh
Hello

# hello.sh
bash: hello.sh: command not found
```

```
# PATH="${PATH}:." hello.sh
Hello
```

. IN ${PATH} 1/4

Aktuelles Verzeichnis mit aufnehmen:

```
export PATH="${PATH}:."  # a) kann man machen
export PATH=".:${PATH}"  # b) STOP!
```

. IN ${PATH} 2/4

```
# ln -s /bin/rm cp
# cp datei{1,2}
```

. IN ${PATH} 3/4

```
# PATH="${PATH}:." cp datei1 datei2
cp: cannot stat 'datei1': No such file or direct

# PATH=".:${PATH}" cp datei1 datei2
cp: cannot remove 'datei1': No such file or dire
cp: cannot remove 'datei2': No such file or dire
```

. IN ${PATH} 4/4

`man bash` → PARAMETERS → Shell Variables → PATH:

A zero-length (null) directory name in the value of PATH indicates the current directory. A null directory name may appear as two adjacent colons, or as an initial or trailing colon.

Beispiele:

```
# export PATH="${PATH}::${HOME}/bin"  # naja
# export PATH="${PATH}:"
# export PATH=":${PATH}"  # STOP!
```

SIGNALE

VORWEG: WAS IST EIN PROZESS?

Ein Prozess ist ein *Programm in Ausführung*.
Er hat einen Zustand, Dateien geöffnet, verbraucht Rechenzeit und Arbeitsspeicher.

DIE WICHTIGSTEN SIGNALE

- 2 = `SIG`**`INT`** — `Ctrl+C`
- 9 = `SIG`**`KILL`**
- 15 = `SIG`**`TERM`**
- 18 = `SIG`**`CONT`**
- (19 = `SIG`**`STOP`**)
- 20 = `SIG`**`TSTP`** — `Ctrl+Z`

(Für mehr: Siehe Ausgabe von `kill -l`)

SENDEN VON SIGNALEN

```
# kill -KILL $(pgrep firefox)
```

oder kürzer

```
# pkill -KILL firefox
  ^
```

JOBS

VORDER- UND HINTERGRUND

By Default: Anwendung bliebt im Vordergrund:

```
# sleep $((5*60))
```

Alternativ: Starten im Hintergrund:

```
# sleep $((5*60)) &
# pid=$!
```

STARTEN IM HINTERGRUND

```
# sleep $((5*60)) &
# pid=$!

# cat -v /proc/${pid}/cmdline
sleep^@300^@

# kill ${pid}
```

Weitere relevante Builtins: `jobs`, `fg`, `bg`

TYPEN: WAS GIBT ES NOCH AUSSER VARIABLEN?

TYPEN

Neben Variablen kennt Bash die Typen:

- **`alias`**
- **`builtin`**
- `file`
- **`function`**
- `keyword`

Ermittelbar mit dem `type`-Builtin:

```
# type -t sleep
file
```

ALIASE

Vorschläge für `~/.bashrc`:

```
alias ls='ls -lhF --color=auto --group-directories-first'
```

```
alias grep='grep --color=auto -I'
```

```
alias ALARM='mplayer -loop 0 /path/to/alarm.mp3 &>/dev/null'
```

(Den Alarm-Sound gibt es als CC-BY lizensiert zum Download auf freesound.org.)

BUILTINS

Ein paar wichtige Shell Builtins:

- `cd`
- `kill`
- `help`
- `history`
- `type`

FUNKTIONEN

Wir könnten regelmäßiges

```
# mkdir projekt19
# cd projekt19
```

abkürzen zu

```
# mkcd projekt19
```

mit einer eigenen Funktion `mkcd`.

STOLPERSTEINE

RM MIT VARIABLEN

```
# rm -Rf "${MISSPELT}"/  # STOP!

# echo "${MISSPELT?}"/
bash: MISSPELT: parameter null or not set

# set -o nounset  # kurz: set -u

# echo "${MISSPELT}"/
bash: MISSPELT: unbound variable
```

SUDO MIT UMLEITUNG 1/2

```
# sudo echo 1234 > /root/datei4
bash: /root/datei4: Permission denied
```

Problem: Umleitung nach `/root/datei4` benötigt bereits Root-Rechte — `sudo` kommt *zu spät*.

SUDO MIT UMLEITUNG 2/2

```
# sudo echo 1234 > /root/datei4
bash: /root/datei4: Permission denied

# sudo bash -c 'echo 1234 > /root/datei4'
```

BEI INLINE-BLÖCKEN ; VOR }

```
hello() {
    echo Hello
}
```

Aber:

```
hello() { echo Hello ; }
                     ^
```

GLOBBING OHNE ERGEBNISSE / NULLGLOB 1/2

```
# for i in *.pdf ; do echo "${i}" ; done ; echo Done.
```

GLOBBING OHNE ERGEBNISSE / NULLGLOB 2/2

```
# for i in *.pdf ; do echo "${i}" ; done ; echo Done.
*.pdf
Done.

# shopt -s nullglob

# for i in *.pdf ; do echo "${i}" ; done ; echo Done.
Done.
```

AUSRUFEZEICHEN

```
# echo 'Error!!!'
Error!!!

# echo {1..3}
1 2 3

# echo "Error!!!"  # History expansion
echo "Errorecho {1..3}!"
Errorecho {1..3}!

# sudo !!  # Vorheriges Kommando, diesmal mit sudo
```

EXPANSION

EXPANSION: ÜBERSICHT 1/2

- Brace Expansion
- Tilde Expansion
- Parameter and Variable Expansion
- Command Substitution
- Arithmetic Expansion
- Process Substitution
- Pathname Expansion
- History Expansion
- Word Splitting, Quote Removal

EXPANSION: ÜBERSICHT 2/2

- Brace Expansion — `DSC_4650.{JPG,jpeg},{1..3}`
- Tilde Expansion — `~/projects/`
- Parameter and Variable Expansion — `${3}`, `${HOME}`
- Command Substitution — `$(pgrep thunderbird)`
- Arithmetic Expansion — `$((5 * 1024**3))`
- Process Substitution — `<(ps aux)`
- Pathname Expansion — `*.txt`
- History Expansion — `!-3`
- Word Splitting, Quote Removal

BRACE EXPANSION

```
# convert ~/Desktop/screenshot19.{png,jpeg}
```

(`convert` ist Teil von ImageMagick® und dessen Fork GraphicsMagick.)

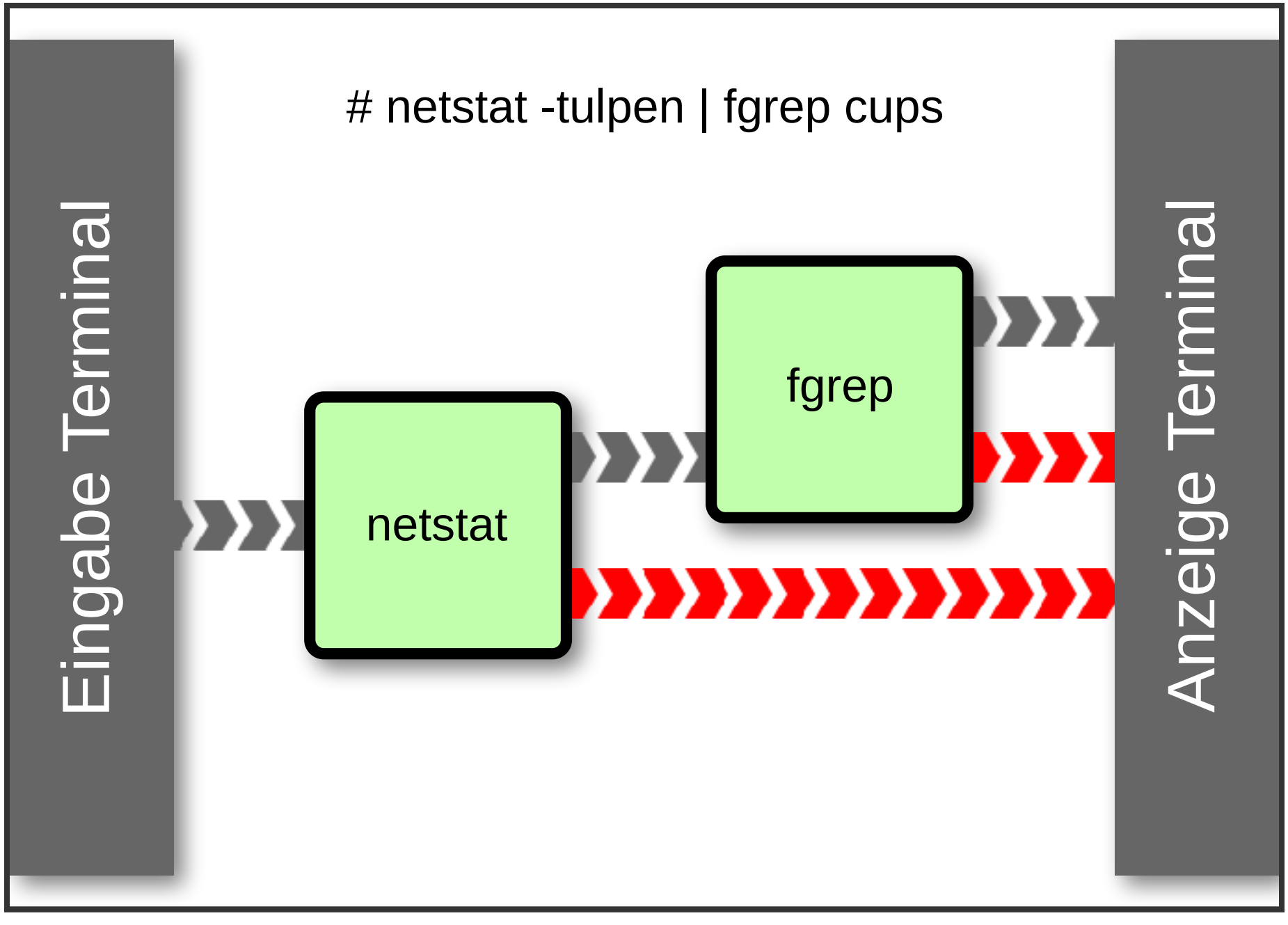
ARITHMETIC EXPANSION

```
# sleep $((5*60))
# sleep 5m
```

```
# truncate --size $((5 * 1024**3)) 5gib.img
# truncate --size 5g 5gib.img
```

```
# dd if=/dev/zero of=5gib.img bs=$((1024**2)) count=$((5*1024))
# dd if=/dev/zero of=5gib.img bs=M count=5k
```

PIPING


netstat -tulpen | fgrep cups
Eingabe Terminal
netstat
fgrep
Anzeige Terminal

TEXT PROCESSING

- `head -n 3`
- `tail -n 4`
- `grep`, `fgrep`
- `awk '{print $2}'`
- `sort`, `sort -u`
- `sed 's,alt,neu,g'`
- `tr`
- `column -t`

MEINE TOP ~15

TOP HOTKEYS

- `Ctrl + C` — `SIGINT`
- `Ctrl + D` — EOF (End of file)
- `Ctrl + Z` — `SIGTSTP`
- `Ctrl + L` — `clear` (Bildschirm "löschen")
- `Alt + .`
- `Ctrl + R` — Inkrementelle Suche

(Alle ohne `Shift`)

FIND UND XARGS 1/3

```
# find -name '*.[ch]' | xargs kate &
```

FIND UND XARGS 2/3

```
# find -name '*.[ch]' -print0 | xargs -0 kate &
                      ^^^^^^^         ^^
```

FIND UND XARGS 3/3

```
# find -type f -name '*.[ch]' -exec kate {} + &
       ^^^^^^^                  ^^^^^^^^^^^^^^^
```

ZWISCHEN ZWEI ORDNERN VOR- UND ZURÜCK WECHSELN:

```
cd -
```

(Siehe auch pushd/popd)

PIPEN AN LESS:

```
.............. | less
```

KOPIE VON AUSGABE UMLENKEN

```
..... |& tee log.txt
```

RETURN-CODE VISUALISIEREN

- `......... ; echo $?`
- `......... && echo GOOD`
- `......... || echo FAILED >&2`
- `export PS1="\${?#0}${PS1}"`

ECHO ALS TESTLAUF VOR DINGE SCHREIBEN

```
for i in ......... ; do
    echo mv "${i}" "${i%.log}.txt"
#   ^^^^
done
```

REKURSIVES GREP

```
# fgrep -R  ....... .......
# fgrep -Rl ....... ....... | xargs sed -i ......
```

BRACE EXPANSION

```
# mv ~/.thunderbird{,_BEFORE}
```

```
# cp datei{,_BACKUP_2015-12-12}
```

XTRACE DEBUGGING AKTIVIEREN

```
set -x
```

oder

```
set -o xtrace
```

RECHNER HERUNTERFAHREN (WENN ETWAS FERTIG IST)

```
sudo sh -c 'while pgrep emerge ; do \
sleep 1 ; done ; poweroff'
```

IN NEUES VERZEICHNIS WECHSELN

```
mkcd() {
    local dir="$1"
    mkdir "${dir}" && cd "${dir}"
}

mkcd projekt19
```

IN NEUES **WEGWERF**-VERZEICHNIS WECHSELN

```
cd "$(mktemp -d)"
```

PROZESSE AUFLISTEN (MIT DETAILS)

```
ps aux | fgrep ........
```

ÜBER WELCHE DIENSTE BIN ICH PER NETZWERK ANGREIFBAR?

```
sudo netstat -tulpen
```

WAS FÜR LOKALE IP-ADRESSEN HAT MEIN SYSTEM?

```
ip addr
```

KONTEXT VON `cd` BESCHRÄNKEN

```
( cd path/to/folder && ......... )
```

TERMINAL REPARIEREN

`<Ctrl+C>`, `reset`, `<Enter>`

SCHREIB-ANFRAGEN PERSISTIEREN

sync

BONUS-TRACK: /USR/BIN/ENV

/USR/BIN/ENV

hat folgende Aufgaben:

1. Umgebungsvariablen listen
2. Programme mit anderer Umgebung aufrufen:
   - `env HOME=/foo ........`
   - `env -i HOME=/foo ........`
3. Pfadauflösung *mit* Ausführung

BEISPIEL-WRAPPER FÜR CP

```
#! /usr/bin/env bash
# Wrapper around cp(1) with invocation echo
PS4='# '
set -x
exec /bin/cp "$@"
```

Diese Folien sind entstanden mit 100% freier Software, konkret mit Hilfe von:

- Asciidoctor
- asciidoctor-reveal.js
- Git
- GNU make
- Inkscape
- reveal.js

auf Gentoo Linux in Kate/KWrite und Yakuake.

SELBST NÄHER ANSEHEN:

- Hardlinks, Softlinks, `/dev/null`
- coreutils — `chroot`, `dd`, `tac`, …
- procps — `pidof`, `watch`, …
- psmisc — `killall`, `pstree`, …
- util-linux — `findmnt`, `losetup`, `lsblk`, …
- `debootstrap`, `git`, `htop`, `kpartx`, `rsync`, `ssh`, `strace`, `tmux`
- `ack`, `hidesvn`, `optipng`, `ncdu`, `pv`, `tree`, `xsel`

FRAGEN?!

Sebastian Pipping

sebastian@pipping.org

https://blog.hartwork.org/

https://github.com/hartwork/