Linux Basics

Jörg Faschingbauer

www.faschingbauer.co.at

jf@faschingbauer.co.at

Table of Contents

FASCHINGBAUER

Overview

FASCHINGBAUER

Overview

FASCHINGBAUER

Zentrale Begriffe

FASCHINGBAUER

- Kernel
- Userspace
- Prozess
- Filedeskriptor
- ... und ein paar mehr

Overview

FASCHINGBAUER

Prozesse

FASCHINGBAUER

- Getrennte Adressräume
- Schutzverletzungen
- Attribute (UID, GID, CWD, ...)
- Ressourcenlimits
- ...

Threads - “Lightweight Processes”

FASCHINGBAUER

- Sind Teil eines Prozesses
- Teilen sich mit dem Prozess einen Adressraum (Fluch und Segen)
- → Synchronisationsmechanismen
- → Kommunikationsmechanismen
- Nicht ursprünglicher Bestandteil von UNIX
- → schiessen ein wenig quer

Scheduling

FASCHINGBAUER

- Kernel verteilt CPU-Ressourcen an Prozesse (und Threads)
- Prozesse und Threads gleichermaßen wichtig
- Traditionell: *faires* Scheduling → keine Garantien, wann wer drankommt.
- Realtime-Optionen; durchaus geeignet für zeitkritische Anwendungen

Overview

FASCHINGBAUER

Das Filesystem

FASCHINGBAUER

Es gibt nur eine einzige Hierarchie, und die beginnt beim Root Directory ('/'), bestehend aus

- Directories
- Files
- Hard- und Softlinks
- Device Special Files
- Erweitert durch Mounts an Mountpoints

Everything is a File

FASCHINGBAUER

- Filedeskriptoren (und Prozesse) sind *das* zentrale Konzept in UNIX
- ... und ganz speziell in Linux

Overview

FASCHINGBAUER

Kernel (1)

FASCHINGBAUER

Stellt sicher, dass es der “Userspace” bequem hat:

- Linearer Adressraum mit Swapspace
- Präemptives Multitasking $\rightarrow$ Gerechtigkeit
- Keine Interrupts, die alles kaputtmachen. Oder doch? $\rightarrow$ Signale!
- Schutz von Individuen gegeneinander
- Hardware ist als solche nicht sichtbar

Kernel (2)

FASCHINGBAUER

Fakten:

- Es gibt keinen Kernel-Prozess! Der Kernel ist die Summe aller Prozesse, die am System laufen, gespickt mit Hardware-Interrupts.
- Ein Prozess wechselt in *Kernel Mode* durch *System Calls*.
- Im Kernel Mode herrschen alle Freiheiten.

Overview

FASCHINGBAUER

User Space

FASCHINGBAUER

Geschützter Bereich, wo die “normalen” Programme leben.

- Eigene, unendlich große Adressräume
- Shell
- C-Library
- Nette Programmierparadigmen, die wir demnächst kennenlernen werden

Overview

FASCHINGBAUER

Nach dem trockenen Zeug ein paar Beispiele

FASCHINGBAUER

Die grundlegenden Konzepte sind eng miteinander verwoben.

- Was wäre ein Prozess ohne *Current Working Directory*?
- Wer soll Files schreiben, wenn nicht ein Prozess?
- Wie wird beim Boot der Userspace geboren? Wer startet den ersten Prozess?
- Wo nähme der Kernel das erste Programm her, wenn nicht aus dem (Root-)Filesystem?
- ...

Da helfen nur Beispiele ...

Overview

FASCHINGBAUER

Die Shell, entmystifiziert (1)

FASCHINGBAUER

Starten eines Programmes, nichtdestruktiv

```
$ sleep 10
...
$
```

Hier passiert Folgendes:

- Shell generiert Child-Prozess und *wartet*, bis er *terminiert*
- Child *exekutiert* `/usr/bin/sleep`
- Child *terminiert*

Die Shell, entmystifiziert (2)

FASCHINGBAUER

Starten eines Programmes, destruktiv

```
$ exec sleep 10
```

Was war das eben?!

Trennung zwischen Prozess und Executable

FASCHINGBAUER

In Windows sind Prozesse an das Ausführen eines Programmes gebunden:

`CreateProcess()` Startet einen neuen Prozess von einem Executable

Unix ist anders:

`fork()` Startet einen neuen Prozess. Selbes Executable, Kopie des Parent-Adressraums.

`exec()` Lädt ein Executable *über* den gerade laufenden Prozess — gegenwärtiger Inhalt wird ersetzt.

Das proc Filesystem

FASCHINGBAUER

Virtuelles Filesystem, das einen Blick ins System bietet. Zum Beispiel:

/proc/self

```
$ ls -l /proc/self
lrwxrwxrwx 1 root root ... /proc/self -> 3736
$ ls -l /proc/self
lrwxrwxrwx 1 root root ... /proc/self/
```

Bitte herumstöbern!

Preisfrage: warum ist `/proc/self/exe` ein Symbolischer Link auf `/bin/ls`?

Executable?

FASCHINGBAUER

Permissions

```
$ ls -l /bin/ls
-rwxr-xr-x 1 root root 109736 Jan 28 18:13 /bin/ls
```

Die Datei heisst nicht `ls.exe`, sondern sie ist *ausführbar* (executable).

Executable: Shared Libraries

FASCHINGBAUER

Shared Libraries

```
$ ldd /bin/ls
 linux-vdso.so.1 =>  (0x00007fff15b69000)
 librt.so.1 => /lib/librt.so.1 (0x00007fa763546000)
 libacl.so.1 => /lib/libacl.so.1 (0x00007fa76333d000)
 libc.so.6 => /lib/libc.so.6 (0x00007fa762fe4000)
 libpthread.so.0 => /lib/libpthread.so.0 (0x00007f...
 /lib64/ld-linux-x86-64.so.2 (0x00007fa76374f000)
 libattr.so.1 => /lib/libattr.so.1 (0x00007fa762bc...
```

Executable: Memory Mappings

FASCHINGBAUER

Virtuelles Memory wird benützt, um das Speicherabbild des Prozesses herzustellen:

/proc/<pid>/maps

```
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:02 1375644 /bin/cat
0060a000-0060b000 r--p 0000a000 08:02 1375644 /bin/cat
0060b000-0060c000 rw-p 0000b000 08:02 1375644 /bin/cat
...
```

Overview

FASCHINGBAUER

Simple is beautiful

FASCHINGBAUER

Um Einfachkeit zu erzielen, muss man oft ein wenig länger nachdenken. Das macht sich danach unendlich oft bezahlt.

Klar: ein File ist ein File

FASCHINGBAUER

Ein File ist ein File. Ok, das ist leicht. Mit Tools, die dafür gemacht sind, Files zu schreiben und zu lesen, eben das zu machen, kann jeder.

Schreiben ins File

```
$ echo Hallo > /tmp/ein-file
```

Lesen vom File

```
$ cat /tmp/ein-file
Hallo
```

Ist eine serielle Schnittstelle ein File?

FASCHINGBAUER

Warum nicht? Daten gehen raus und kommen rein!

Schreiben ins Kabel

```
$ echo Hallo > /dev/ttyUSB0
```

Lesen vom Kabel

```
$ cat /dev/ttyUSB1
Hallo
```

Pseudo Terminals

FASCHINGBAUER

- Historisch: Login über Hardware-Terminal, verbunden über serielle Leitung
- Terminal (TTY) Layer (im Kernel) implementiert Session Management
- *Pseudo Terminal*: Software statt Kabel

Analog zur Ausgabe über serielles Kabel:

Schreiben auf ein Pseudo-Terminal

```
$ echo Hallo > /dev/pts/0
```

Disks und Partitionen

FASCHINGBAUER

USB Stick Backup

```
# cat /proc/partitions
major minor  #blocks  name

   8        32    2006854 sdc
   8        33    2006823 sdc1
# cp /dev/sdc1 /Backups/USB-Stick
# mount -o loop /Backups/USB-Stick /mnt
```

/proc und /sys

FASCHINGBAUER

- Kernel hat Variablen (im Memory), die gewisse Aspekte seiner Aufgaben konfigurieren
- Die meisten dieser Variablen werden als Files angeboten

Corefiles sollen core.<PID> heissen

```
# echo core.%p > /proc/sys/kernel/core_pattern
```

Suspend to Disk

```
# echo disk > /sys/power/state
```

Zufallszahlen

FASCHINGBAUER

Der Kernel bzw. Driver sammeln Entropie aus dafür geeigneten Quellen (Interrupts).

Entropie-Pool leeren

```
$ cat /dev/random
```

Overview

FASCHINGBAUER

Overview

FASCHINGBAUER

Commandline: Aufbau (1)

FASCHINGBAUER

Die Commandline besteht meistens (alles nur Konvention!) aus drei Teilen:

- Command: ausführbares Programm (oder Script), Alias, oder Shell-Builtin
- Optionen: z.B. `-i` ("Short Option"), oder `--interactive` ("Long Option")
- Argumente: meist Dateinamen

Commandline: Aufbau (2)

FASCHINGBAUER

Beispiel: cp

```
$ cp -i -r /etc ~/tmp
$ cp -ir /etc ~/tmp
```

- Command ist `cp`
- Optionen sind `-i` und `-r`
- Argumente sind `/etc` und `~/tmp`

Commandline: Optionen

FASCHINGBAUER

Alternativ: "Long Options"; z.B. bei `cp`

- `--interactive` statt `-i`
- `--recursive` statt `-r`

"Standard" Optionen:

- `-h` oder `--help`: kurze Erläuterung des Commands
- `-v` oder `--verbose`: Ausführliche Meldungen
- `--version`: Version des Commands

Ausreisser:

dd

```
$ dd if=/dev/zero of=/dev/null count=20 bs=1024
```

Hilfe und Manual (1)

FASCHINGBAUER

Befehl erklärt seine Optionen mit `--help`

`--help`

```
$ cp --help
```

Vollständige Beschreibung in der Manual Page ("Man-Page")

Manual Pages

```
$ man cp
```

Hilfe und Manual (2)

FASCHINGBAUER

Navigation in der Anzeige von Man-Pages

h	Hilfe von `man`
q	Beendet `man` (oder die Hilfe)
g	Springt zum Anfang
G	Springt zum Ende
/	Sucht vorwärts im Text
?	Sucht rückwärts im Text
n	Springt zur nächsten Fundstelle
N	Springt zur vorherigen Fundstelle

Hilfe für `man` selbst

```
$ man man
```

Line Editing (1)

FASCHINGBAUER

Geniale Editiermöglichkeiten (Emacs-Keys!)

← , →	Cursor links, rechts
⟵ (Backspace)	Löscht das Zeichen links vom Cursor
Entf , Strg + d	Löscht das Zeichen unter dem Cursor
Pos 1 , Strg + a	Springt zum Zeilenanfang
Ende , Strg + e	Springt zum Zeilenende
Strg + k	Löscht vom Cursor bis zum Zeilenende
Strg + u	Löscht vom Cursor bis zum Zeilenanfang
Strg + l	Löscht den Bildschirm bis auf die aktuelle Zeile

Line Editing (2)

FASCHINGBAUER

Wortweises Editieren

Strg + ← , Alt + b	Springt ein Wort nach links
Strg + → , Alt + f	Springt ein Wort nach rechts
Alt + d	Löscht Wort rechts vom Cursor
Alt + ⟵	Löscht Wort links vom Cursor

Anmerkungen

- Strg + ← und Strg + → funktionieren nicht auf der virtuellen Konsole (belegt mit Konsolen-Switch)
- Sollte Alt nicht belegt sein, nimmt man Esc

Line Editing (3)

FASCHINGBAUER

Keys für Kenner, Liebhaber und Emacs-Benutzer

Strg + y , Alt + y	Einfügen aus dem "Kill-Ring"
Strg + _ (Underline)	Undo
Strg + t	Zeichen tauschen ("transpose")
Alt + t	Wörter tauschen
Alt + u	Wort in Uppercase wandeln
Alt + l	Wort in Lowercase wandeln
Alt + c	1. Buchstabe groß ("capitalize")

Tabulatortaste (1)

FASCHINGBAUER

- Tabulatortaste Spart Tastendrücke (kein Mensch kann Maschinschreiben)
- Commandvervollständigung

Alle Commands, die mit `git` beginnen

```
$ git 
git                    git-p4                 git-resurrect          gi
git-cvsserver          git-receive-pack       git-shell              gi
```

Tabulatortaste (2)

FASCHINGBAUER

- Filenamenvervollständigung

Piepen, falls nicht eindeutig

```
$ less /var/log/emerge-log/emerge ⟺
```

Zwei Tabs → Liste

```
$ less /var/log/emerge-log/emerge ⟺  ⟺
emerge-fetch.log  emerge.log
```

Tabulatortaste (3)

FASCHINGBAUER

- Eigene Vervollständigungen mit `complete`

Eigene Vervollständigungen

```
$ complete -W "server laptop" ssh
$ ssh s ⇐⇒
```

Wird zu "`ssh server`"
Dauerhaft eingetragen in `~/.bashrc`

Overview

FASCHINGBAUER

Variablen

FASCHINGBAUER

Shell ist eine vollständige Programmiersprache → es gibt Variablen

- Shell-Variablen: nur gültig in der Shell, in der sie definiert wurden
- Environment-Variablen: werden vererbt (→ inhärentes Konzept von Unix)

(Shell-Scripts von unerfahrenen Programmierern benutzen oft nur Environment-Variablen, weil der Unterschied nicht bekannt ist → "globale Variablen")

Shell-Variablen

FASCHINGBAUER

Shell-Variable

```
$ HISTSIZE=1000
$ echo $HISTSIZE
1000
```

- Bei Zuweisung/Definition sind keine Spaces erlaubt
- Dereferenziert mit $
- Shell-Variablen sind nur im Kontext der aktuellen Shell gültig

Shell-Variablen - Gültigkeit

FASCHINGBAUER

Shell-Variable wird nicht vererbt

```
$ A_SHELL_VARIABLE=value
$ echo $A_SHELL_VARIABLE
value
$ bash
$ echo $A_SHELL_VARIABLE

$ exit     # oder Ctrl-D
$ echo $A_SHELL_VARIABLE
value
```

Environment-Variablen

FASCHINGBAUER

Shell-Variablen werden mit `export` zu Environment-Variablen → vererbt über Prozessgrenzen

Environment-Variable

```
$ export A_SHELL_VARIABLE
$ bash
$ echo $A_SHELL_VARIABLE
value
```

Environment-Variablen sind Konzept von Unix; die Shell versucht, sie gleich aussehen zu lassen wie Shell-Interne Variablen.
→ `/proc/<pid>/environ`

Variablen: $HOME

FASCHINGBAUER

$HOME ist reine Konvention, um es Shell-Scripts einfacher zu machen.

- Home-Directory definiert in /etc/passwd (→ struct passwd, man getpwnam)
- Login-Programm (/bin/login, sshd, gdm, ...) liest /etc/passwd und setzt $HOME
- Login-Programm startet Login-Shell oder Desktop
- → $HOME ist für alle Prozesse des Benutzers definiert

man bash dokumentiert $HOME als normale Shell-Variable (→ wird von Bash ja nicht gesetzt)

Variablen: `$PATH`

FASCHINGBAUER

- Liste von Directories (durch ':' getrennt), in denen die Shell nach Executables sucht.
- Shell sucht nicht in Current Working Directory → Sicherheitsrisiko (ausser man hat '.' in `$PATH`)
- Eigene Directories vorne dran, z.B.
  `export PATH=$HOME/bin:$PATH`

Variablen: $PS1, $PS2

FASCHINGBAUER

- `$PS1`: Primärer Shell-Prompt; sichtbar wenn die Shell ein Command erwartet.
- `$PS2`: Sekundärer Shell-Prompt; sichtbar, wenn die Shell mehr Eingabe braucht, um das Command zu beenden.
- Normale Shell-Variablen (keine Environment-Variablen)
- Viele Spezialzeichen mit besonderer Bedeutung $\rightarrow$ `man bash`

Weitere bekannte Variablen

FASCHINGBAUER

- `$EDITOR`: Name des eingestellten Editier-Programms. Z.B. Subversion poppt `$EDITOR` auf, um die Commit-Message eingeben zu lassen.
- `$PAGER`: Programm, das zum Seiten-Blättern genommen wird. Z.B. von `man`.
- `$TMPDIR`: Ort für temporäre Files
- `$LD_LIBRARY_PATH`: Suchpfad für Shared Libraries, die in Nicht-Standard-Directories liegen
- `$PYTHON_PATH`: wie `$LD_LIBRARY_PATH`, nur für Python-Module

Overview

FASCHINGBAUER

History

FASCHINGBAUER

Wiederholtes Tippen ist unnötig → History

History

```
$ history
...
  620  ssh 192.168.1.104
  621  ping 192.168.1.104
  622  ssh -Y 192.168.1.104
$ !620
ssh 192.168.1.104
```

History: Aufruf

FASCHINGBAUER

Weitere Aufrufmöglichkeiten

History

```
$ !ss
ssh -Y 192.168.1.104
$ !!
ssh -Y 192.168.1.104
$ !-2
ping 192.168.1.104
```

History: Navigieren

FASCHINGBAUER

Tastenkombinationen (teilweise wieder an Emacs angelehnt)

Strg + p , ↑	Ein Item rauf
Strg + n , ↓	Ein Item runter

Inkrementelle Infix-Suche rückwärts: unverzichtbar!

- Zum Beispiel Strg + r `h -Y` (Infix eines Commands aus der History) → `ssh -Y 192.168.1.104`
- Übernehmen des Kommandos durch Enter oder durch eine der Editierkombinationen.

History: Command, Variablen

FASCHINGBAUER

Das `history`-Command

`-c`	Löschen der History
`-d <num>`	Löschen eines Commands

- (Mehr → `man history`)

Variablen

`HISTSIZE`	Anzahl der gemerkten Commands (default 500)
`HISTFILE`	History-File (default `~/.bash_history`)
`HISTFILESIZE`	Anzahl der in `HISTFILE` gespeicherten Commands (defaul

- (Mehr → `man bash`)

Overview

FASCHINGBAUER

Aliases

FASCHINGBAUER

Viele Commands haben viele Optionen, die man nicht jedesmal tippen will
→ Command "Aliases".

Zum Beipiel:

- `alias d='ls -al'`; lange Directory Listings
- `alias cp='cp -i'`; Promptet vor Überschreiben
- `alias rm='rm -i'`; Promptet vor Löschen (nervig)

Aufruf des orginalen Commands:
`$ \rm /etc/passwd`
Definieren von Aliases am besten in `~/.bashrc`

Overview

FASCHINGBAUER

Bash: Übungen

FASCHINGBAUER

- Aus den bisher getippten Commands finden Sie mittels inkrementeller Suche den heraus, der Ihnen am besten gefiel.
- Welche Option muss man bei `echo` angeben, damit am Ende des Textes kein Linefeed ausgegeben wird?
- Finden Sie heraus, wie man das `cp` Command dazu bewegen kann, symbolische Links zu dereferenzieren und als Files zu behandeln.
- Definieren Sie (in `~/.bashrc` einen Alias Ihrer Wahl für den Aufruf von `ls`, der Ihnen am besten entspricht. (`man ls`; Kandidaten: `ls -l` oder `ls -al`)
- Modifizieren Sie Ihren primären Shell-Prompt so, dass auch Username, Current Working Directory und Uhrzeit angezeigt werden. (Hinweis: `man bash`, suchen nach "PROMPTING")

Overview

FASCHINGBAUER

Overview

FASCHINGBAUER

Absolute und relative Pfade

FASCHINGBAUER

Relative Pfade

- Beziehen sich auf das *Current Working Directory* (CWD) des Prozesses
- Beginnen *nicht* mit / oder ~

Absolute Pfade

- Beziehen sich auf das Root-Directory (/)

Spezielle Pfade

FASCHINGBAUER

Spezielle Pfade

- /: Root Directory
- ~: Home Directory des eingeloggten Benutzers (von der Shell interpretiert, *nicht* vom Kernel - der weiss nichts von Home Directories)
- `~jfasch`: Home Directory des Benutzers `jfasch` (von der Shell interpretiert)
- `.`: Current Working Directory (vom Kernel interpretiert)
- `..`: Parent Directory des CWD (vom Kernel interpretiert)

Sonstige Besonderheiten

FASCHINGBAUER

Sonstige Besonderheiten

- `.` und `..` sind Einträge im Filesystem und können Teile von Pfaden sein.
- `~jfasch/../bheide` → Home Directory meines Nachbarn.
- `~jfasch/.` → äquivalent zu `~jfasch`.
- Die Einzelheiten — was wird von der Shell und was vom Kernel interpretiert — erkennt man deutlich, wenn man eine alternative Shell (z.B. Busybox `sh`) verwendet.

Overview

FASCHINGBAUER

Directories

FASCHINGBAUER

Commands

`cd`	Wechseln des CWD
`pwd`	"Print Current Working Directory"
`mkdir`	Neues Directory anlegen
`rmdir`	(Leeres) Directory löschen
`rm -r`	Directory rekursiv löschen
`mv`	File oder Directory verschieben

cd

FASCHINGBAUER

`$ cd /`	Ins Root Directory
`$ cd home/jfasch`	Umständlich von dort ins Home Directory
`$ cd ..`	Eine Ebene rauf
`$ cd .`	Nop
`$ cd`	Einfach ins Home Directory
`$ cd ~`	Umständlich ins Home Directory
`$ cd -`	Zum Directory, in dem man zuletzt war

mkdir

FASCHINGBAUER

Neues Directory in einem existierenden Directory

mkdir

```
$ mkdir /tmp/a-directory
```

Neues Directory, samt komplettem Pfad

mkdir -p

```
$ mkdir /tmp/an/entire/hierarchy
mkdir: cannot create directory ‘/tmp/an/entire/hierarchy’: No
$ mkdir -p /tmp/an/entire/hierarchy
```

rmdir

FASCHINGBAUER

Löschen eines leeren Directorys

rmdir

```
$ rmdir /tmp/a-directory
```

Löschen eines nicht-leeren Directorys

rm -r

```
$ rmdir /tmp/an
rmdir: failed to remove ‘/tmp/an’: Directory not empty
$ rm -r /tmp/an
```

Overview

FASCHINGBAUER

Symbolische Links

FASCHINGBAUER

Symbolische Links

- Sehen aus wie andere Directory-Einträge (Directories, Files, ...)
- In Wirklichkeit nur eine Referenz auf einen anderen Eintrag
- Ziel muss nicht existieren → "Broken Link"
- Fehlerhafe Links können Zyklen ergeben → Tools verfolgen Links by Default *nicht* (`cp --dereference ...`)
- Werden vom Kernel interpretiert (Doze: "Shortcuts"; vom Explorer interpretiert)

Symbolische Links: Commands

FASCHINGBAUER

Commands

- Symbolischen Link anlegen
  `ln -s TARGET LINK`
- Syntax ähnlich wie `cp` oder `mv`
- Symbolischen Link löschen
  `rm` (gilt für alle Directory-Einträge)

Symbolischer Link: Typ

```
$ ls -l /usr/bin/vi
lrwxrwxrwx 1 root root ... /usr/bin/vi -> vim
```

Hardlinks

FASCHINGBAUER

Hardlinks

- Zweiter Eintrag für ein Filesystemelement → nicht zu unterscheiden vom ursprünglichen Element
- Kreise schwerwiegender als bei Symlinks → Hardlinks auf Directories (und andere Hardlinks) verboten
- Nur von `root` anzulegen

Hardlink: Typ

```
# ln /usr/bin/vi /usr/bin/vi-hardlink
# ls -l /usr/bin/vi-hardlink
-rwxr-xr-x 2 root root ... /usr/bin/vi-hardlink
```

Overview

FASCHINGBAUER

Current Working Directory (1)

FASCHINGBAUER

- Grundlegende Eigenschaft eines Prozesses

pwd

```
$ pwd
/home/jfasch
$ strace pwd
...
getcwd("/home/jfasch", 4096) = 13
...
```

→ `pwd` ist also ein normales Programm, das von der Shell gestartet wird, das CWD vererbt bekommt, selbiges ausgibt, und terminiert.

Current Working Directory (2)

FASCHINGBAUER

Ist cd auch ein normales Programm?

cd

```
$ strace cd
strace: cd: command not found
$ type cd
cd is a shell builtin
```

(Klar eigentlich?)

Overview

FASCHINGBAUER

Directory Listings (1)

FASCHINGBAUER

Listing des CWD

```
$ ls
Archiv  Desktop  Documentation
```

Listing eines beliebigen Directory

```
$ ls /tmp
ssh-ISxKsBgO2151  virtual-jfasch.LpAlcl  orbit-jfasch
```

Directory Listings (2)

FASCHINGBAUER

Haufenweise Optionen, auszugweise:

`-l`	langes Ausgabeformat
`-a`	Ausgabe von "versteckten" Einträgen (".")
`-d`	Directory, nicht Inhalt
`-h`	"Human Readable"
`-t`	Sortierung nach Timestamp
`-R`	Rekursiv mit Subdirectories

Typ in der ersten Spalte

`-`	File
`d`	Directory
`c`	"Character Special Device"
`b`	"Block Device"
`l`	Symlink

Attribute

FASCHINGBAUER

Attribute von Filesystemelementen

- Creation Time
- Access Time (ausser bei `noatime` Mount-Option)
- Modification Time

Manipulation durch `touch`:

- `touch file`: Anlegen eines neuen Files
- `touch -a file`: Access Time modifizieren
- `touch -m file`: Modification Time modifizieren

Nicht von `ls` angezeigt; statt dessen:

- `stat file`

Overview

FASCHINGBAUER

Kopieren (cp) (1)

FASCHINGBAUER

Sicherheitskopie von `.bashrc`**, in** `~`**:**

```
$ cp .bashrc .bashrc.save
```

Kopieren von `.bashrc` **nach** `/etc/bash/bashrc`**:**

```
$ cp .bashrc /etc/bash/bashrc
```

Kopieren von `.bashrc` **nach** `~/tmp`**:**

```
$ cp .bashrc ~/tmp
```

(Wenn `~/tmp` nicht existiert, endet `.bashrc` als `~/tmp`)

Kopieren mehrerer Files:

```
$ cp /media/stick/*.jpg ~/Archiv/Bilder/Urlaub
```

(Zieldirectory muss existieren)

Kopieren (cp) (2)

FASCHINGBAUER

Optionen (auszugsweise):

`-r`	rekursiv inklusive Subdirectories
`-p`	Timestamp und Permissions werden erhalten
`-v`	Verbose
`-i`	Prompt, bevor was überschrieben wird

Verschieben (mv)

FASCHINGBAUER

Konzeptuell wie Kopieren und Löschen

- Es werden nur Einträge umgehängt (atomar)
- Ursprünglich nur innerhalb desselben Filesystems (`rename()` System Call)
- Verschieben zwischen Filesystemen ist "Kopieren und Löschen" $\rightarrow$ langsamer
- Verschieben von Directories von Natur aus rekursiv $\rightarrow$ keine `-r` Option

Overview

FASCHINGBAUER

Owner und Permissions

FASCHINGBAUER

Typen von Berechtigungen

- Lesen (`r`)
- Schreiben (`w`)
- Ausführen (`x`)

Separate Berechtigungen für

- Benutzer (`u`)
- Gruppe (`g`)
- Andere (`o`)

Permission Bits

FASCHINGBAUER

File Permissions

```
$ ls -l /etc/passwd
-rw-r--r-- ... /etc/passwd
```

Bits	Bedeutung
`-`	Typ: Reguläre Datei
`rw-`	Schreib- und lesbar für Owner (root)
`r--`	Lesbar für Gruppe
`r--`	Lesbar für Andere

Execute Permissions

FASCHINGBAUER

Execute Permissions

```
$ ls -l /bin/ls
-rwxr-xr-x ... /bin/ls
```

→ Eine ausführbare Datei muss nicht mit `.exe` enden. Sie muss ausführbar sein.

Directory Permissions

FASCHINGBAUER

Directory Permissions

```
$ ls -ld /etc
drwxr-xr-x ... 07:54 /etc
```

- Read Permissions: der Inhalt (Liste der Namen) darf gelesen werden.
- Execute Permissions: um ein File z.B. lesen zu dürfen, muss man Execute Permissions am Parent Directory und allen Directories am Pfad haben

Permission Bits, oktal

FASCHINGBAUER

`ls -l` Output	Binär	Shell-Command
`-rw-r--r--`	110100100	`chmod 0644 ...`
`-rw-------`	110000000	`chmod 0600 ...`
`-rwxr-xr-x`	111101101	`chmod 0755 ...`

System Calls verlangen einen Integer → meist oktal angegeben.

Default Permissions – umask

FASCHINGBAUER

U-Mask (Bitfeld) wird von Berechtigungen abgezogen. U-Mask ist eine vererbte Prozesseigenschaft.

umask in Action

```
$ umask
0022
$ touch /tmp/file
$ ls -l /tmp/file
-rw-r--r-- ... /tmp/file
```

umask: Wie geht das?

FASCHINGBAUER

- U-Mask (Bitfeld) wird von Berechtigungen abgezogen
- U-Mask ist eine vererbte Prozesseigenschaft.
- Ausgangsbasis beim Anlegen eines neuen Files: `rw-rw-rw-`

Ausgangspermissions	`rw-rw-rw-`	110 110 110	0666
- U-Mask	`----w--w-`	000 010 010	0022
Ergibt	`rw-r--r--`	110 100 100	0644

Shell Commands

FASCHINGBAUER

- Ändern von Permissions (oktales Setzen):
  `$ chmod 755 ~/bin/script.sh`
- Ändern von Permissions (differenziell symbolisch):
  `chmod u+x,g-wx,o-rwx ~/bin/script.sh`
- Ändern der Gruppe (nur als root bzw. Angehöriger der Gruppe):
  `chgrp audio /tmp/file`
- Ändern des Users (nur als root):
  `chown user /tmp/file`
- `chmod`, `chown`, und `chgrp` interpretieren `-R` als "rekursiv".

Set-UID Bit

FASCHINGBAUER

Motivation, z.B.: Passworte stehen verschlüsselt in `/etc/passwd` bzw. `/etc/shadow` → nur `root` darf schreiben. Damit User `jfasch` sein Passwort ändern kann, muss er `root` werden, ohne das Rootpasswort zu haben:

passwd

```
$ ls -l /bin/passwd
-rws--x--x 1 root root ... /bin/passwd
```

Sticky Bit

FASCHINGBAUER

In /tmp dürfen alle schreiben. Wo kämen wir hin, wenn sich die User gegenseitig die Files wegnehmen könnten?

Sticky Bit an /tmp

```
$ ls -ld /tmp
drwxrwxrwt ... /tmp
```

Owner und Permissions: System Calls

FASCHINGBAUER

man 2 chown

```
int chown(const char *path, uid_t owner, gid_t group);
int fchown(int fd, uid_t owner, gid_t group);
int lchown(const char *path, uid_t owner, gid_t group);
```

man 2 chmod

```
int chmod(const char *path, mode_t mode);
int fchmod(int fd, mode_t mode);
```

Overview

FASCHINGBAUER

find: Grundlegendes

FASCHINGBAUER

find geht ein Directory rekursiv durch und schreibt (ohne Optionen) alle Einträge auf stdout.

Alle Einträge unterhalb des CWD

```
$ find
```

Alle Einträge unterhalb eines beliebigen Directorys

```
$ find /etc/init.d
```

Alle Einträge unterhalb einer Liste von Directories

```
$ find /etc/init.d /tmp
```

`find`: Optionen

FASCHINGBAUER

Durch Optionen werden gefundene Einträge gefiltert.
Unübliche Syntax: *Optionen kommen am Ende!*
Gebräuchliche Optionen:

`-name` *name*	Einträge mit Namen *name*
`-type` *typ*	Einträge mit Typ *typ* (`f`, `d`, `c`, `b`, ...)
`-user` *user*	Einträge, die User *user* gehören
`-group` *group*	Einträge, die Gruppe *group* gehören
`-mmin` *num*	Einträge, die *genau num* Minuten alt sind
`-mmin -`*num*	Einträge, die *höchstens num* Minuten alt sind
`-exec`	Aufrufen eines Commands pro gefundenem Eintrag

find: Beispiele

FASCHINGBAUER

Alle Einträge unter /etc**, die** net **heissen**

```
$ find /etc -name net
```

Alle Einträge unter /etc**, die mit** net **beginnen.** find **versteht Shell-Globs, aber Vorsicht: Quoting nicht vergessen!**

```
$ find /etc -name 'net*'
$ find /etc -name net\*
```

Alle Subdirectories in meinem Homedirectory:

```
$ find ~-type d
```

Alle Entries in /var**, die jünger als 10 Minuten sind:**

```
$ find /var -mmin -10
```

Alle Headerfiles des Kernels in einer Wurscht:

```
$ find /usr/src/linux/ -name \*.h -exec cat {} \;
```

find: Verknüpfungen

FASCHINGBAUER

Kriterien können mit folgenden Operatoren verknüpft werden

- -a: AND
- -o: OR
- !: NOT; Vorsicht, Quoting: History Expansion
- (und): Klammerung; Vorsicht, das sind Shell-Metacharacters → Quoting

find: Verknüpfungen: Beispiele

FASCHINGBAUER

Alle Einträge unter /var, die root gehören und jünger als 30 Minuten sind:

```
$ find /var -user root -a -mmin -30
```

Alle Files in meinem Homedirectory, die jünger als 30 Minuten oder größer als 50K sind:

```
$ find ~-type f -a \( -mmin -30 -o -size +50k \)
$ find ~-type f -a '(' -mmin -30 -o -size +50k ')'
```

1001 weitere Optionen → man find

Overview

FASCHINGBAUER

Filesystem: Übungen (1)

FASCHINGBAUER

- Wechseln Sie nach `/tmp` und legen Sie dort ein Directory `parent` und in diesem ein Directory `child` an (wenn geht mit einem einzigen Command). Wechseln Sie in Ihr Home Directory und löschen Sie von dort aus diese beiden Directories mit einem einzigen Command.
- Legen Sie zwei symbolische Links an, die aufeinander zeigen. Was geschieht, wenn Sie einen von ihnen dereferenzieren (z.B. mit `cat`)?

Filesystem: Übungen (2)

FASCHINGBAUER

- Kopieren Sie Ihr Home Directory nach `/tmp/my-home` (und löschen es wieder).
- Bewundern Sie `/dev` mit Hilfe eines "langen Listings", und interpretieren Sie die verschiedenen Eintrags-Typen.
- Suchen Sie in `/var` alle Files, die älter als eine Stunde und kleiner als 50k sind.
- Legen Sie ein File `-rf` an (z.B. mit `echo > -rf`) und löschen Sie es wieder.

Overview

FASCHINGBAUER

Overview

FASCHINGBAUER

`stdin`, `stdout`, und die Pipe

FASCHINGBAUER

- Philosophie von Unix ist "Jedes Tool tut genau eine Sache, und das gut" (oder so ähnlich)
- Kombination von Tools mittels *Pipe* → Output eines Tools (*Standard Output* — `stdout`) wird als Input eines anderen Tools (*Standard Input* — `stdin`) genommen
- Viele Files (Konfiguration, Logfiles, Source) sind traditionell textbasiert → viele verschiedene Tools, die Textfiles behandeln

stdin, stdout, und die Pipe

FASCHINGBAUER

Anzahl der #includes im Kernel Source

```
$ find /usr/src/linux/ -name '*.[hc]' -exec cat {} \;| \
   grep '^#include'|wc -l
```

In welchen Gruppen bin ich?

```
$ grep jfasch /etc/group|cut -d: -f1
# oder auch:
$ id jfasch
```

Grundlegende Tools

FASCHINGBAUER

Grundlegende Tools

`cat`	Ausgeben eines Files nach `stdout`
`head`	Ausgeben der ersten paar Zeilen
`tail`	Ausgeben der letzten paar Zeilen
`cut`	Ausschneiden von Feldern und Zeichen
`less`	"Pagen" eines Files
`sort`	Zeilenweise Sortieren
`uniq`	Gleiche Zeilen eliminieren (nach `sort`)
`grep`	Zeilen nach Regular Expressions filtern

Overview

FASCHINGBAUER

cat

FASCHINGBAUER

- Kurzform für Con**cat**enate — "zusammenhängen"
- `cat file1 file2 ...`: gibt Files hintereinander auf `stdout` aus

cat

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
daemon:x:2:2:daemon:/sbin:/bin/false
...
```

Overview

FASCHINGBAUER

head und tail

FASCHINGBAUER

Die ersten 5 Zeilen von `/etc/passwd`**:**

```
$ head -n 5 /etc/passwd
```

Alles ausser den letzten 5 Zeilen von `/etc/passwd`**:**

```
$ head -n -5 /etc/passwd
```

Die letzten 5 Zeilen von `/etc/passwd`**:**

```
$ tail -n 5 /etc/passwd
```

Neues aus `/var/log/messages`**:**

```
$ tail -f /var/log/messages
```

Overview

FASCHINGBAUER

cut

FASCHINGBAUER

Gibt ausgewählte Felder/Teile jeder Zeile eines Files aus.

Alle Gruppen, die im System definiert sind

```
$ cut -d :  -f 1 /etc/group
```

Alle Gruppen, die im System definiert sind, mit ihren IDs

```
$ cut -d :  -f 1,3 /etc/group
```

Overview

FASCHINGBAUER

Durchblättern von Files mit less

FASCHINGBAUER

- Gibt Text Seite für Seite aus
- Vor- und Zurückblättern, Suchen möglich
- Bedienung wie bei man (less ist der Default-"Pager" von man)

Tastenkombinationen:

h	Hilfe
q	Beendet man (oder die Hilfe)
g	Springt zum Anfang
G	Springt zum Ende
/	Sucht vorwärts im Text
?	Sucht rückwärts im Text
n	Springt zur nächsten Fundstelle
N	Springt zur vorherigen Fundstelle

Overview

FASCHINGBAUER

Files durchsuchen mit grep

FASCHINGBAUER

- **G**lobal **R**egular **E**xpression **P**rint: aus sed (**S**tream **ED**itor) Terminologie
- Alltägliches Tool: Zeitwort "greppen"

Alle Vorkommnisse von jfasch in /etc/group

```
$ grep jfasch /etc/group
adm::4:root,adm,daemon,jfasch
portage::250:portage,jfasch
jfasch:x:1000:
```

grep: Regular Expressions

FASCHINGBAUER

Regular Expressions

- Feingranularere Selektionsmöglichkeiten als ein Wort
- Quoting meist nötig, um Shell Expansion zu verhindern

jfasch, als Mitglied einer Gruppe

```
$ grep '^.*[:,]jfasch' /etc/group
adm::4:root,adm,daemon,jfasch
portage::250:portage,jfasch
```

grep: Optionen

FASCHINGBAUER

Nützliche Optionen für grep

`-i`	Groß/Kleinschreibung ignorieren
`-l`	Zeigt den Dateinamen ohne Treffer
`-n`	Zeilennummern
`-r`	Directories rekursiv
`-v`	Nur Zeilen, die *nicht* passen

Regular Expressions: mehr Info

FASCHINGBAUER

Regular Expression sind ähnlich lustig wie Programmieren, geben aber einen eigenen Kurs her. Mehr Information:

- `man grep`
- `man perlre`, `man perlrequick`, `man perlretut`
- Jeffrey E.F. Friedl: Mastering Regular Expressions (O'Reilly); http://regex.info/

Overview

FASCHINGBAUER

Tools für Textfiles: Übungen

FASCHINGBAUER

1. Zeigen Sie die ersten beiden Zeilen von `/etc/passwd` an.
2. Geben Sie alle Gruppen im System samt ihrer ID aus (siehe dazu `/etc/group`).
3. Finden Sie durch Greppen heraus, welche Portnummer das File Transfer Protokoll (FTP) verwendet. Hinweis: Portnummern (Alt-Unixisch: "Services") sind in `/etc/services` registriert. ($\rightarrow$ `man services`)
4. Verwenden Sie `less` für dieselbe Aufgabe.
5. Zeigen Sie die dritte Zeile von `/etc/passwd` an (Hinweis: verwenden Sie `head` und `tail` in einer Pipe).

Overview

FASCHINGBAUER

Overview

FASCHINGBAUER

Standard I/O Streams

FASCHINGBAUER

- Drei Standard I/O *Filedescriptoren*
- Alle drei sind an das Terminal gekoppelt

Bedeutung	Name	Descriptor-Nummer	C-Macro
Standard Input	`stdin`	0	`STDIN_FILENO`
Standard Output	`stdout`	1	`STDOUT_FILENO`
Standard Error	`stderr`	2	`STDERR_FILENO`

Religiös eingehaltene Tradition unter Unix:

- Debug/Error-Output geht nach *Standard Error* und *nicht* nach Standard Output. Standard Output ist für die Pipe da.
- Programme sind nur in Ausnahmefällen nicht für die Pipe gedacht.

I/O Redirection

FASCHINGBAUER

Redirection mit Hilfe der Shell:

`command < file`	`command` bekommt `file` auf Standard Input
`command > file`	`command` schreibt Standard Output auf `file`
`command 2> file`	`command` schreibt Standard Error auf `file`

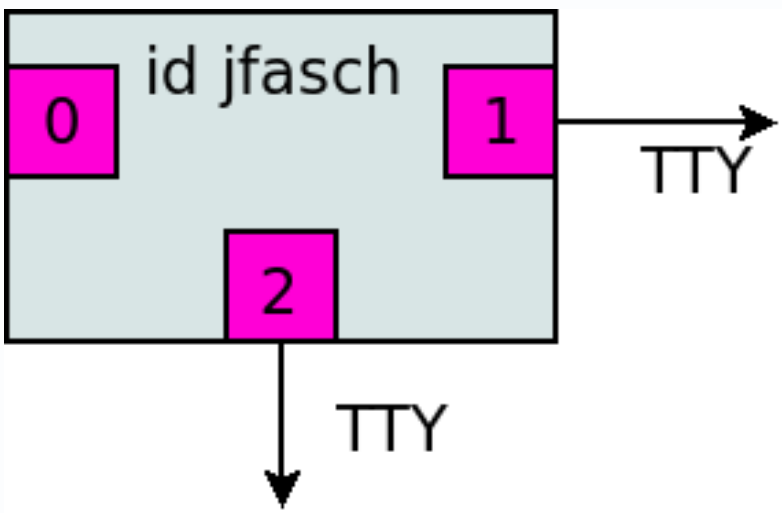
- '>' löscht den Inhalt des Files vorher, falls es existiert
- → '>>', um anzuhängen

Keine Redirection

FASCHINGBAUER

Keine Redirection

```
$ id jfasch
uid=1000(jfasch) gid=1000(jfasch) groups=...
```

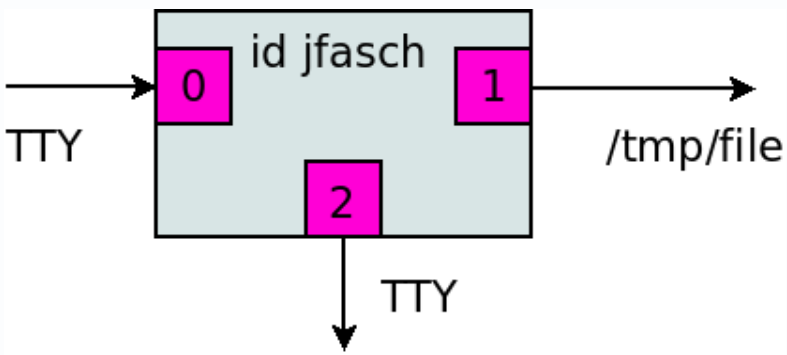


Output Redirection

FASCHINGBAUER

Output Redirection

```
$ id jfasch > /tmp/file
# oder ...
$ id jfasch 1> /tmp/file
```

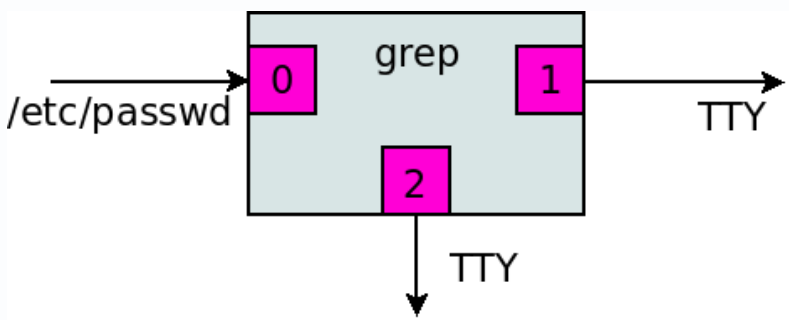


Input Redirection

FASCHINGBAUER

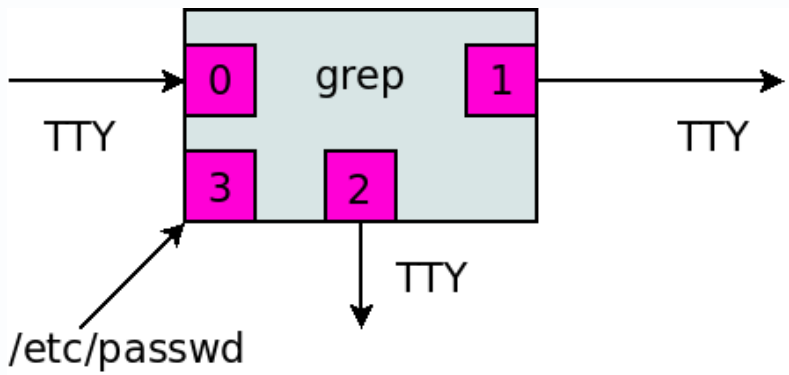
Input Redirection

```
$ grep jfasch < \
    /etc/passwd
```



grep ohne Redirection

```
$ grep jfasch /etc/passwd
```



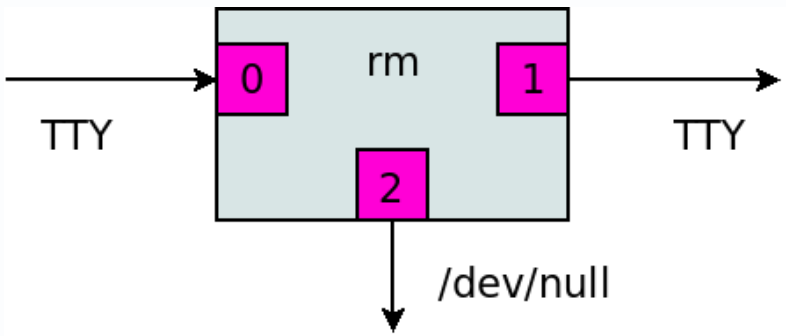
Error Redirection

FASCHINGBAUER

Fehlermeldungen unterdrückt man so ...

Error Redirection

```
$ rm -f /etc/passwd 2> /dev/null
```

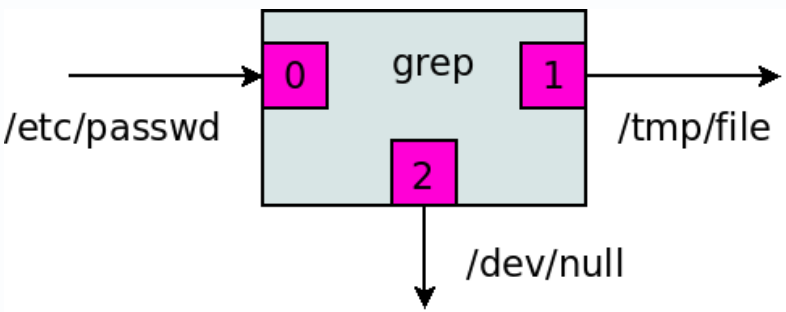


Alles Redirection

FASCHINGBAUER

Alles Redirection

```
$ grep jfasch < /etc/passwd > /tmp/file 2> /dev/null
```



Jonglieren (1)

FASCHINGBAUER

Programme, die auf `stdout` schreiben, sind gute Mitglieder einer Pipe ...

Zählen der Einträge in `/etc`

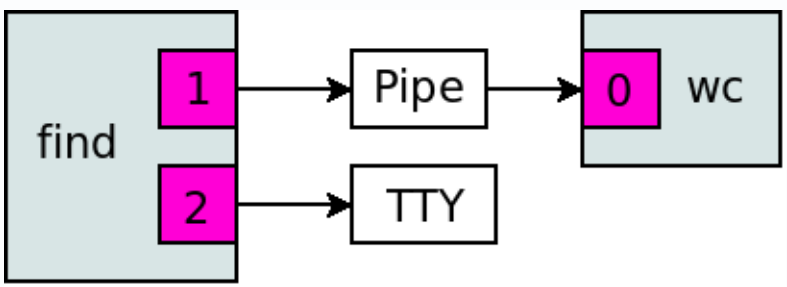
```
$ find /etc | wc -l
find: ‘/etc/cron.daily’: Permission denied
find: ‘/etc/sudoers.d’: Permission denied
find: ‘/etc/cron.weekly’: Permission denied
...
1558
```

Jonglieren (2)

FASCHINGBAUER

Aktionen der Shell:

- Alloziert Pipe ($\rightarrow$ `man 2 pipe`)
- Dupliziert `stdout` von `find` auf den Input der Pipe
- Dupliziert `stdin` von `wc` auf den Output der Pipe ($\rightarrow$ `man 2 dup`)



Jonglieren (3)

FASCHINGBAUER

Problem: wie zählt man Fehlermeldungen?
→ Vertauschen von `stdout` und `stderr`

Zählen der Fehlermeldungen

```
$ find /etc 3>&1 1>&2 2>&3 | wc -l
```

Häh?!

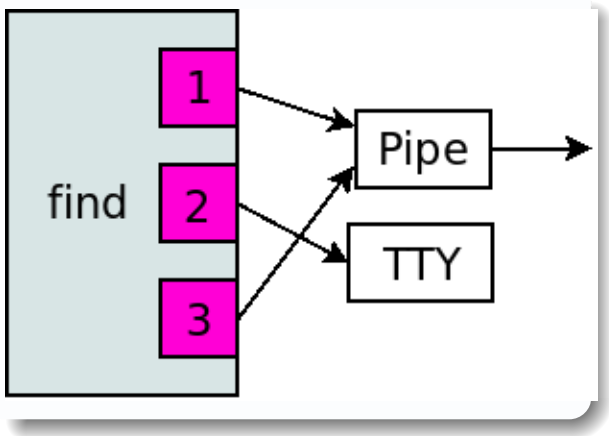
- Evaluierung von links nach rechts (`3>&1` vor `1>&2` ...)
- `A>&B` heisst: "Mach, dass Filedeskriptor A auf das gleiche zeigt wie Filedeskriptor B!"

Jonglieren (4)

FASCHINGBAUER

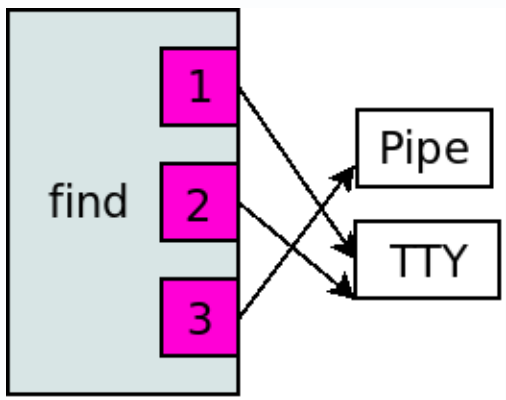
`3>&1`

Wegspeichern von `stdout` nach 3



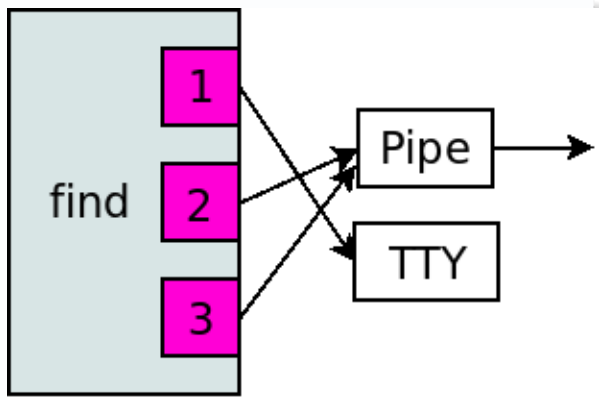
`1>&2`

`stderr` nach `stdout` zeigen lassen



`2>&3`

`stderr` mit Ex-`stdout` überschreiben



- Optional: Schliessen von 3: `3>&-`

Overview

FASCHINGBAUER

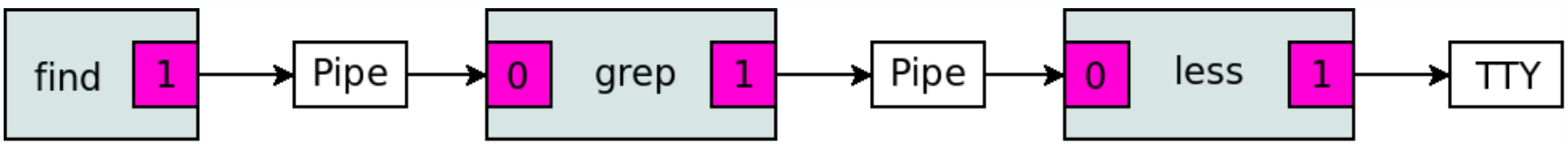
Pipes (1)

FASCHINGBAUER

- Weiterer UNIX-Leitsatz: jedes Werkzeug soll eine Sache machen, und das gut
- Pipe kombiniert Werkzeuge

Alle Kernel-Files, die nicht compiliert werden

```
$ find /usr/src/linux/|grep -v \*.c|less
```



Pipes (2)

FASCHINGBAUER

- *Keine* temporären Files involviert (unter Doze schon)
- Kommunikationsmechnismus
- Pipe = Buffer von beschränkter Größe
- Linker und rechter Prozess arbeiten *gleichzeitig* ($\rightarrow$ Scheduling)
- Schreibender Prozess wird suspendiert, wenn Pipe voll
- Lesender Prozess wird suspendiert, wenn Pipe leer

Pipes: Beispiele (1)

FASCHINGBAUER

Alle User im System, alphabetisch

```
$ cat /etc/passwd|cut -d : -f 1|sort
# effizienter:
$ cut -d : -f 1 < /etc/passwd|sort
```

..., mit Gruppen und IDs

```
$ cut -d : -f 1 < /etc/passwd| \
    sort | \
    while read user; do id $user; done
```

Pipes: Beispiele (2)

FASCHINGBAUER

..., zwischengespeichert in `/tmp/users.txt`

```
$ cut -d : -f 1 < /etc/passwd| \
    sort | \
    tee /tmp/users.txt | \
    while read user; do id $user; done
```

`tee` ... *T-Stück*, als Verbinder zwischen zwei Rohren (Pipes)

Named Pipes

FASCHINGBAUER

- Rendezvous wie (unnamed) Pipe, nur im Filesystem
- Wird behandelt wie ein File, aber: Inhalt nicht persistent
- → `man mkfifo`, `man 3 mkfifo`, `man 7 fifo`

In einem Terminal ...

```
$ mkfifo /tmp/fifo
$ echo Hallo > /tmp/fifo
```

In einem anderen Terminal ...

```
$ cat /tmp/fifo
```

Overview

FASCHINGBAUER

IO-Redirection, Pipes: Übungen (1)

FASCHINGBAUER

- Kopieren Sie mit Hilfe von `cat` (ohne Argumente) `/etc/passwd` nach `/tmp`.
- Erstellen Sie mittels mehrerer Aufrufe von `echo` ein File, das Ihren Namen und Ihre Adresse enthält.
- Geben Sie eine sortierte Liste aller User in Ihrem System aus (`/etc/passwd` enthält alle User samt deren Einzelheiten, durch ':' getrennt).
- Leiten Sie `stdout` *und* `stderr` des Commands `find /etc` nach `/tmp/output` um.
- Warum ist das (nichtleere) File `/tmp/output` nach dem Command `cat < /tmp/output > /tmp/output` leer?
- Erklären Sie den Effekt des Commands (`/tmp/output` ist wiederum nicht leer) `cat < /tmp/output >> /tmp/output`!

IO-Redirection, Pipes: Übungen (2)

FASCHINGBAUER

- Wieviele Directories enthält Ihr Homedirectory? (Hinweis: kombinieren Sie `find` und `wc` mit einer Pipe)
- Statten Sie das Command `find /etc | wc` mit geeigneten Umleitungen aus, sodass die Anzahl der Fehler gezählt wird und `stdout` von `find` nach `/dev/null` geht.
- Schicken Sie unter Zuhilfenahme von `echo`, `cat` und `mkfifo` das Wort "Hallo" im Kreis herum.

Overview

FASCHINGBAUER

Pipes: System Calls (1)

FASCHINGBAUER

Shell verwendet *Systemcalls* wie jeder andere:

- `fork()` bzw. `clone()`: Erzeugen eines neuen Prozesses
- `dup2()`: Duplizieren eines Filedescriptors ($\rightarrow$ vgl. "IO Redirection")
- `execve()` ($\rightarrow$ man 3 exec): Ersetzen des Prozesslayouts aus einem Executable File

Wir gewinnen den "Useless Use of `cat` Award", um zu sehen, wie die Shell die Pipe implementiert (`http://partmaps.org/era/unix/award.html`)

```
$ strace -f bash -c 'cat /etc/passwd|grep jfasch'
```

Pipes: System Calls (2)

FASCHINGBAUER

Die Shell in Aktion: Pipe

```
6313 pipe([3, 4]) = 0
6313 clone(child_stack=0,flags=...) = 6314
6313 close(4) = 0
6313 clone(child_stack=0,flags=...) = 6315
6314 close(3) = 0
6314 dup2(4,1) = 1
6314 close(4) = 0
6313 close(3) = 0
6315 dup2(3,0) = 0
6315 close(3) = 0
6314 execve("/bin/cat",["cat","/etc/passwd"],...) = 0
6315 execve("/bin/grep",["grep","jfasch"],...) = 0
```

Overview

FASCHINGBAUER

Overview

FASCHINGBAUER

Archivierung und Komprimierung: Überblick

FASCHINGBAUER

- UNIX-Leitsatz: jedes Werkzeug soll eine Sache machen, und das gut
- → Komprimieren ist eine Sache, Archivieren eine andere
- Archivierungstools
  - `tar` — wird zumeist verwendet
  - `cpio` — Archivformat von Linux `initramfs` ("because `tar` is ugly as hell" — Al Viro)
- Komprimierungtools. Arbeiten allesamt gut mit der Pipe zusammen. Weitgehend optionskompatibel.
  - `gzip`; Übliche Endung `.gz`
  - `bzip2`; Übliche Endung `.bz2`
  - `lzma`; Übliche Endung `.lzma`

Vergleich von Komprimierungtools

FASCHINGBAUER

Komprimieren eines Tarfiles des Linux-Kernels, ~430MB:

Tool	Verbrauchte Zeit	Resultierende Größe
`gzip`	20s	95MB
`bzip2`	1m 11s	74MB
`lzma`	5m 32s	64MB

Overview

FASCHINGBAUER

gzip: Aufruf

FASCHINGBAUER

Optionen

`-d`	*D*ekomprimieren (default: komprimieren)
`-c`	Output nach `stdout`

Assoziierte Tools:

- `gunzip`. Äquivalent zu `gzip -d`.
- `zgrep`. In komprimiertem File greppen.
- `zcat`. Äquivalent zu `gzip -cd`.
- `zless`. Pagen eines komprimierten Files.

gzip: Beispiele

FASCHINGBAUER

Komprimieren des Files cd.iso. Das File "verwandelt" sich in cd.iso.gz

```
$ gzip cd.iso
```

Komprimieren des Files cd.iso, wobei das Original intakt bleibt

```
$ gzip -c cd.iso > cd.iso.gz
```

Dekomprimieren des Files, mit Zerstörung des Originals

```
$ gzip -d cd.iso.gz
```

Dekomprimieren des Files, ohne Zerstörung des Originals

```
$ gzip -cd cd.iso.gz > cd.iso
```

Overview

FASCHINGBAUER

tar: Grundlegendes

FASCHINGBAUER

- Archiviert Files und Directories ...
- ... und auch sonst alle Eintragstypen ($\rightarrow$ UNIX-spezifisch)
- komprimiert nicht selbst $\rightarrow$ Pipe mit Kompressionstools

Einpacken

```
$ tar -c -f file.tar directory
```

Auspacken

```
$ tar -x -f file.tar
```

Vorschau

```
$ tar -t -f file.tar
```

tar: Arbeitsweise

FASCHINGBAUER

tar packt die Input-Directories so ein, wie man sie angibt ...

Einpacken und Anzeigen

```
$ cd /usr/src/linux
$ tar cf /tmp/filesystems.tar Documentation/filesystems
$ tar tf /tmp/filesystems.tar
Documentation/filesystems/
Documentation/filesystems/ceph.txt
Documentation/filesystems/gfs2.txt
Documentation/filesystems/quota.txt
...
```

tar: Komprimierung

FASCHINGBAUER

Kann gefälligkeitshalber auch komprimieren. Die folgenden Aufrufe sind äquivalent:

Komprimieren

```
$ tar -c -f file.tar.gz -z directory
$ tar -zcf file.tar.gz directory
$ tar cf - directory | gzip > file.tar.gz
```

Dekomprimieren

```
$ tar zxf file.tar.gz
$ tar ztf file.tar.gz
$ gzip -cd file.tar.gz | tar tf -
```

tar: Anstandsregeln

FASCHINGBAUER

- Vorsicht: tar entpackt ins Current Working Directory → vorher unbedingt schauen, was drin ist ...
  `$ tar ztf file.tar.gz`
- Sei nett: tar Files mit mehr als einem Toplevel-Eintrag sind nicht nett! (→ Toplevel soll immer *ein Directory* sein)
- Nimmt keine absoluten Pfade mit (→ weise Vorsicht) ...

Absolute Pfade

```
$ tar cf /tmp/jfasch.tar /home/jfasch
tar: Removing leading ‘/’ from member names
...
```

tar: Optionen

FASCHINGBAUER

Optionen

c	"**c**reate" Archivfile
x	"e**x**tract" Archivfile
t	"**t**ell" Inhalt
f	Von/nach File
v	Verbose
vv	Mehr Verbose
z	gzip anwenden
j	bzip2 anwenden
C	"**C**hange" Directory, bevor Aktion beginnt
T	Fileliste (statt Input-Directory)

Es gibt tausende Optionen (alles, was man braucht)

→ `tar --help`

→ `man tar`

tar: Kreativ in der Pipe (1)

FASCHINGBAUER

Inhalt eines Directory in das andere

```
$ tar -cf - -C ein-directory . | \
      tar -xf - -C anderes-directory
```

Inhalt eines Directory in das andere am Server

```
$ tar -jcf - -C ein-directory . | \
      ssh server 'tar -jxf - -C anderes-directory'
```

tar: Kreativ in der Pipe (2)

FASCHINGBAUER

Selektives kopieren (Fileliste von stdin)

```
$ ( cd /usr/src/linux; \
      find -name \*.h -o -name \*.c | \
      tar -cf - -T - ) | \
    tar -xf - -C anderes-directory
```

Overview

FASCHINGBAUER

gzip, bzip2 und tar: Übungen

FASCHINGBAUER

1. Packen Sie den Inhalt des /etc Directory (ohne /etc selbst) Ihres Computers nach /tmp/etc.tar.bz2, ohne Ihr Current Working Directory zu verlassen.

Overview

FASCHINGBAUER

Overview

FASCHINGBAUER

netstat: Überblick

FASCHINGBAUER

Zeigt Zustand des Netzwerkstacks

- Welche Verbindungen sind offen? Von welchen Programmen?
- Wieviele Pakete/Bytes werden gerade empfangen/gesendet?
- Welche Server-Sockets gibt es?
- TCP- und UNIX-Domain Sockets

netstat: Optionen

FASCHINGBAUER

netstat: Optionen

-a	Auch Server-Sockets (default: nur “established” Verbindungen)
-n	Nur numerische Adressen (keine lange Auflösung zu Namen)
-t	Nur TCP
-u	Nur UDP
-x	Nur UNIX-Domain Sockets
-p	PID und Programmname

Beispiele

FASCHINGBAUER

Alle TCP-Verbindungen (numerisch, incl. Servers, incl. Programmname)

```
$ netstat -antp
```

Erklärung von TCP States:
http://userpages.umbc.edu/~jeehye/cmsc491b/lectures/tcpstate/s

Alle TCP-Server

```
$ netstat -lntp
```

Alle UNIX-Domain Sockets

```
$ netstat -nxp
```

Overview

FASCHINGBAUER

nc: Überblick

FASCHINGBAUER

- Integrieren von Netzwerkverbindungen in die Pipe
- → konsequent, genial, einfach

Zwei Modi

- Client
- Server

nc: Aufruf

FASCHINGBAUER

- **Client**
  `$ nc server.home.com 3456`
  Öffnet aktiv eine Verbindung nach `server.home.com` auf Port 3456.
- **Server (auf `server.home.com`)**
  `$ nc -l -p 3456`
  - Hört auf eingehende Verbindungen
  - Akzeptiert *genau eine* → Pipe

Auch UDP möglich → `-u`

Overview

FASCHINGBAUER

Netzwerken: Übungen

FASCHINGBAUER

- Etablieren Sie eine Netzwerk-Pipe zwischen zwei Prozessen am lokalen Rechner (Hostname `localhost`, Port beliebig aber grösser als 1024) wie folgt
  - Die linke Seite empfängt vom User Zeilen auf `stdin`
  - Die rechte Seite der Pipe (der Server) soll die Anzahl der Zeilen zählen, die der User eingibt
- Inspizieren Sie die Situation mit `netstat`

Overview

FASCHINGBAUER

Overview

FASCHINGBAUER

Prozesse und Scheduling: Überblick

FASCHINGBAUER

Ein Prozess ...

- ist ein Ablauf eines Programms (eines Programm-Files)
- hat einen Stack
- hat eine eindeutige ID (PID — Process ID)
- hat einen Parent-Prozess (ausser `init` mit der PID 1)
- hat ein Current Working Directory (CWD)
- hat offene Files
- hat dynamisch alloziertes Memory ("Heap")
- ...

Beim Tod eines Prozesses werden Ressourcen automatisch vom Kernel "geerntet".

Baumstruktur: PID und PPID

FASCHINGBAUER

Wichtige Tatsachen:

- Alle Prozesse sind in einer Baumstruktur angeordnet
- Root ist `init` — PID 1
- Jeder Prozess kennt seinen Parent
- Ein Prozess wird bei dem Tod eines Kindes (“Child Process”) vom Kernel benachrichtigt (Signal: `SIGCHLD`)
- Beim Tod des Parent wird `init` der neue Parent

Overview

FASCHINGBAUER

ps: Überblick

FASCHINGBAUER

- "**P**rocess **S**tate"
- Zeigt einen Snapshot der Prozesse im System
- GNU ps implementiert mehrere Standards → verwirrende Optionssyntax
- Hier nur "UNIX Standard" (im Gegensatz zu "BSD")
- Am besten durch Beispiele erklärt

Beispiele (1)

FASCHINGBAUER

Default: Prozesse mit gleichem "Controlling Terminal"

```
$ ps
  PID TTY          TIME CMD
  866 pts/5    00:00:00 ps
 6771 pts/5    00:00:00 bash
$ # ... und deren Baumstruktur
$ ps --forest
  PID TTY          TIME CMD
 6771 pts/5    00:00:00 bash
 1086 pts/5    00:00:00  \_ ps
```

Beispiele (2)

FASCHINGBAUER

-e: alle Prozesse

```
$ ps -e
... viele ...
```

-f: mehr Info

```
$ ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
jfasch      1563  6771  0 18:42 pts/5    00:00:00 ps -f
jfasch      6771  2513  0 Sep07 pts/5    00:00:00 -bash
```

Beispiele (3)

FASCHINGBAUER

-F, -l: noch mehr Info (?)

```
$ ps -F
$ ps -Fl
$ ps -fFl
```

Prozesse des Users jfasch

```
$ ps -U jfasch
$ ps -U jfasch -Fl
$ ps -U jfasch -Fl --forest
```

Beispiele (4)

FASCHINGBAUER

-L: Threads

```
$ ps -U jfasch|grep firefox
14080 ?        00:24:17 firefox
$ ps -Lf 14080
UID     ...   LWP  C NLWP ... CMD
jfasch ... 14080  1   12 ... /usr/bin/firefox
jfasch ... 14082  0   12 ... /usr/bin/firefox
...
```

- `LWP` ... "**L**ight**W**eight **P**rocess" (Thread ID). Vom Scheduler wie Prozess behandelt → `strace -p THREAD-ID`
- `NLWP` ... Anzahl der Threads im Prozess

Beispiele (5)

FASCHINGBAUER

-o: User-defined

```
$ ps -o pid,cmd,nlwp
```

Wichtige Identifier:

%cpu	CPU Auslastung
pid	Process ID
ppid	Parent Process ID
args	Komplettes Command (incl. Argumentvektor)
lwp	Thread ID
sched	Scheduling Policy (Realtime → später)
wchan	Punkt im Kernel, an dem der Prozess blockt

Beispiele (6)

FASCHINGBAUER

Sortieren nach CPU-Verbrauch (aufsteigend)

```
$ ps --sort=%cpu -o %cpu,cmd -e
%CPU CMD
 0.0 [kthreadd]
...
 9.5 /usr/lib/nspluginwrapper/i386/linux/npviewer.bin
```

Sortieren nach CPU-Verbrauch (absteigend)

```
$ ps --sort=-%cpu -o %cpu,cmd -e
```

Overview

FASCHINGBAUER

/proc (1)

FASCHINGBAUER

Das (virtuelle) proc Filesystem enthält unter anderem ein Directory für jeden Prozess im System:

/proc

```
$ ls -l /proc
dr-xr-xr-x 7 root     root     ... 16:53 1
dr-xr-xr-x 7 jfasch   jfasch   ... 10:41 12532
dr-xr-xr-x 7 root     root     ... 16:53 1258
dr-xr-xr-x 7 jfasch   jfasch   ... 16:53 14080
dr-xr-xr-x 7 postgres postgres ... 16:53 16123
lrwxrwxrwx 1 root     root     ... 15:13 self -> 32667
...
```

Prozesse: /proc (2)

FASCHINGBAUER

Jedes /proc/PID Directory gibt Einsicht in die dahinterliegende Kernelstruktur für Prozesse → man sieht dort alle Attribute eines Prozesses:

/proc/PID (auszugsweise)

```
$ ls -l /proc/14080
-r--r--r-- ... cmdline
-rw-r--r-- ... comm
lrwxrwxrwx ... cwd -> /home/jfasch
-r-------- ... environ
lrwxrwxrwx ... exe -> /usr/lib64/firefox/firefox
-r--r--r-- ... wchan
```

Prozesse: /proc (3)

FASCHINGBAUER

Weitere bemerkenswerte Attribute

- /proc/14080/fd: benützte Filedescriptoren
- /proc/14080/fdinfo: mehr Info darüber
- /proc/14080/task: Threads
- /proc/14080/root: Prozesse können verschiedene Rootdirectories haben

/proc/self: Symlink, der auf das Directory des lesenden Prozesses zeigt.

Overview

FASCHINGBAUER

Mehr Prozess-Tools ...

FASCHINGBAUER

`ps` bezieht seine Information aus `/proc`. Das tun auch andere ...

- `pidof`: alle Process IDs zu einem Namen
- `top`: periodische Liste aller Prozesse, by Default sortiert nach CPU-Auslastung
- `lsof`: "List Open Files" — welcher Prozess hat welche Files offen?

top: Prozessmonitor (1)

FASCHINGBAUER

- Periodisches Display von Prozessen — by Default fallend sortiert nach CPU-Auslastung.
- Am besten ausprobieren ...

h	**Hilfe**
< / >	Sortierspalte nach links/rechts
x	Sortierspalte hervorheben
f	Felder hinzufügen
o	Reihenfolge ändern
z	Running Tasks hervorheben ($\rightarrow$ Multiprozessor)
d	Update Intervall ändern (Default: 3 Sekunden)
H	Threads anzeigen
W	**Konfiguration schreiben** (`~/.toprc`)

top: Prozessmonitor (2)

FASCHINGBAUER

Commandline-Optionen ...

Nur Prozesse 1305, 14652 und 72349:

```
$ top -p 1305,14652,72349
```

Nur meine Prozesse:

```
$ top -u jfasch
```

Nur 3 Iterationen, dann selbst beenden:

```
$ top -n 3
```

Arbeitsweise begutachten ...

```
$ strace top
```

`lsof`: List Open Files (1)

FASCHINGBAUER

Probleme mit Files, Directories, etc.:

- Man darf Files löschen, auch wenn sie noch benutzt werden → u.U. will man schauen, ob sie jemand benutzt (z.B. unbenutzte Sockets in `/tmp`)
- Warum kann ich das Device nicht unmounten? (Wer benutzt es noch?)
- Wer hat diese TCP-Connection nach ... offen?
- Laufen gerade Prozesse von diesem Executable?
- Wer benutzt diese Shared Library gerade?

`lsof`: List Open Files (2)

FASCHINGBAUER

Alle offenen Files im System:
`$ lsof`
Wer benutzt `/var/log/messages`**?**
`# lsof /var/log/messages` → Klar: `syslog` (als `root`)
Wer hat eine HTTP-Verbindung nach `www.google.com`**?**
`$ lsof -i @www.google.com` → 3mal `firefox`

Wichtig: alle Pfade müssen absolut angegeben werden → `lsof` sucht einfach in `proc`, und dort stehen sie so drin!

Overview

FASCHINGBAUER

Background-Prozesse (1)

FASCHINGBAUER

Die Shell blockiert, wenn ein Command beim Arbeiten ist → "Vordergrund" (Foreground)

- Programm bekommt *Signale* vom *Terminal* zugestellt (Strg - C bzw. `SIGINT`, ...)
- Programm bekommt `stdin` vom Terminal
- Shell wartet auf Programm-Ende
- Shell liest nicht von `stdin`, schreibt nicht auf `stdout` oder `stderr`

Background-Prozesse (2)

FASCHINGBAUER

Gegenteil: "Hintergrund" (Background)

- Shell bleibt achtsam → führt weitere Commands aus
- Shell merkt sich den Hintergrund-Prozess → "Job"
- Terminal-Keys gehen an die Shell
- `stdin` ist für Background-Prozesse tabu → `SIGTTIN` bei Lesen → Prozess wird *suspendiert*

Background-Prozesse (3)

FASCHINGBAUER

Prozess im Background starten

```
$ xclock&
[1] 12295
```

- `[1]` ... "Job-ID". Ein reines Shell-Konstrukt → hat nichts mit Kernel zu tun.
- Shell merkt sich die Background-Prozesse als Jobs
- Referenziert als %1
- z.B. `kill %1` — *fast* äquivalent zu `kill 12295`

Background-Prozesse (4)

FASCHINGBAUER

Killen des Jobs

```
$ xclock&
[1] 12295
$ kill %1
[1]+  Terminated                 xclock
```

→ Shell meldet sofort “Terminated”

Background-Prozesse (5)

FASCHINGBAUER

Killen des Prozesses

```
$ xclock&
[1] 12646
$ kill 12646
$ # return
[1]+  Terminated                 xclock
```

(Eigentlich nicht so wichtig fürs wirkliche Leben)

Background-Prozesse (6)

FASCHINGBAUER

Foreground-Prozess in den Background stellen

```
$ xclock
^Z
[1]+  Stopped                 xclock
$ bg  # oder auch "bg 1"
[1]+ xclock &
```

- Strg + z : Terminal Driver schickt `SIGTSTP` an den Prozess
- Prozess wird *suspendiert*
- `bg` registriert ihn als *Job* und setzt ihn *im Background* fort (mittels `SIGCONT`)

Overview

FASCHINGBAUER

Signale (1)

FASCHINGBAUER

"Events" mit Nummer 1 bis 31, die ein Prozess einem anderen schickt. Hauptsächlich von der Shell bzw. dem Kernel ausgeschickt, um Prozessen etwas mitzuteilen — zum Beispiel ...

- Nette Bitte zur Beendigung
- Brutales Abschießen
- Fehler
- ...

Signale (2)

FASCHINGBAUER

Gemeinhin bekannte Signale

`SIGINT`	Strg + c , Bitte zur Beendigung, abfangbar
`SIGQUIT`	Strg + \ , Bitte zur Beendigung, abfangbar
`SIGTERM`	`kill PID`, Bitte zur Beendigung, abfangbar
`SIGKILL`	`kill -9 PID`, brutal, nicht abfangbar
`SIGSTOP`	Anhalten, nicht abfangbar
`SIGTSTP`	Anhalten (Strg + z), abfangbar
`SIGCONT`	Fortsetzen, nicht abfangbar
`SIGFPE`	Floating Point Exception, abfangbar
`SIGHUP`	"Hangup" (historisch) $\rightarrow$ Config-Reload

Signale (3)

FASCHINGBAUER

Liste aller verfügbaren Signale, incl. Nummern

```
$ kill -l
 1) SIGHUP  2) SIGINT  3) SIGQUIT  4) SIGILL  ...
 6) SIGABRT  7) SIGBUS  8) SIGFPE  9) SIGKILL ...
...
```

Nettes Beenden von Prozessen

```
$ kill -TERM 1234
$ kill -SIGTERM 1234
$ kill -15 1234
```

Signale (4)

FASCHINGBAUER

- Fehlersignale `SIGSEGV`, `SIGFPE`, ... werden vom Kernel geschickt
- Können auch mit `kill` verteilt werden → Boshaftigkeit Kollegen gegenüber

Mehr Information:

- `man 7 signal`
- Programmierer Achtung: Ausschau halten nach dem Wort "*async-signal-safe*"!

Overview

FASCHINGBAUER

Scheduling

FASCHINGBAUER

- "Scheduler" verteilt Rechenzeit an Prozesse und Threads
- "Time Sharing": alle kommen mal dran
- "Context Switch": Prozesswechsel
- Scheduling ist transparent: Prozesse merken nichts davon
- "Präemptives Multitasking": keine Kooperation von Programmen notwendig (zum Unterschied von Windows 3.11, Mac OS $<$ X, verschiedenen Embedded "OS")
- Ein Thread bekommt gleich viel wie ein Prozess
- *Keine* garantierten Antwortzeiten

Fairness

FASCHINGBAUER

- UNIX-Tradition: *fair!*
- "IO-Bound" Prozesse: nützen ihren Timeslice nicht aus (warten meist auf IO)
- "Compute-Bound" Prozesse: rechenintensiv
- IO-Bound Prozesse kommen schneller dran (Punktesystem) $\rightarrow$ bessere Reaktionszeit für User (Maus und Tastatur!)

Nice-Value

FASCHINGBAUER

- Nice-Value (Prozess-Attribut) beeinflusst die Zuteilung der Rechenzeit
- Werte zwischen -20 (gar nicht nett) und +19 (ganz nett — wird nur laufen, wenn sonst keiner will)
- Default: 0
- Einen Prozess weniger nett machen kann nur `root`

Prozess mit Nice-Value 10 starten

```
$ nice -n 10 sha1sum /dev/sda1
```

Laufenden Prozess um 7 netter machen

```
$ renice -n +7 16569
```

Realtime

FASCHINGBAUER

- Krasser Gegensatz zur traditionellen UNIX-Welt → *unfair*
- Klar geregelte Prioritäten
- Ist ein Prozess *runnable*, unterbricht er einen mit geringerer Priorität
- Garantierte Antwortzeiten (na ja, fast)

Bisher unbekannte Gefahren:

- Endlos-Loop mit Realtime-Priorität
- Deadlocks (Priority Inversion)
- → eine weitere Dimension von Fehlern

Scheduling Policies

FASCHINGBAUER

- Traditionelles (faires) UNIX-Scheduling: `SCHED_OTHER`

Realtime Schduling Policies (evtl. von Embedded OSs her bekannt)

- `SCHED_FIFO`: Prozess läuft so lange, bis er von einem mit höherer Priorität unterbrochen wird (oder von selbst schlafen geht)
- `SCHED_RR` (Round Robin): Prozesse mit gleicher Priorität wechseln innerhalb kurzer Timeslices. Ansonsten wie `SCHED_FIFO`

Scheduling Prioritäten

FASCHINGBAUER

- Traditionelle (`SCHED_OTHER`) Prozesse leben in der guten alten Priorität 0
- Realtime-Prioritäten: 1-99

Anzeige der Realtime-Prioritäten

```
$ ps -o ...,rtprio,...
```

Scheduling Prioritäten

FASCHINGBAUER

- Traditionelle (`SCHED_OTHER`) Prozesse leben in der guten alten Priorität 0
- Realtime-Prioritäten: 1-99

Anzeige der Realtime-Prioritäten

```
$ ps -o ...,rtprio,...
```

Scheduling: Tools

FASCHINGBAUER

Ausführen eines Prozesses mit Fifo Scheduling und Priorität 42

```
chrt -f 42 sleep 7
```

Anheben der Priorität und der Policy eines existieren Prozesses

```
chrt -p -f 42 4697
```

Overview

FASCHINGBAUER

Prozesse und Scheduling: Übungen (1)

FASCHINGBAUER

- Wie viele Prozesse laufen auf Ihrem System?
- Welcher Prozess braucht am meisten Memory?
- Warum zeigt `ls -l /proc/self` jedesmal auf ein anderes Directory, wo es doch auf die gegenwärtig laufende Shell-Instanz zeigen sollte?
- Wie kann man nur mit Filesystem-Mitteln die PID der gerade laufenden Shell herausfinden?
- Finden Sie den Prozess in Ihrem System, der am meisten Threads hat!

Prozesse und Scheduling: Übungen (2)

FASCHINGBAUER

- Interpretieren Sie den Effekt des Commands
  `$ chrt -f 1 sha1sum /dev/zero`
  (als `root`)
- Gibt es auf Ihrem System Prozesse mit Realtime Priorität? Was tun sie?

Overview

FASCHINGBAUER

Overview

FASCHINGBAUER

Was bisher geschah: Commandline

FASCHINGBAUER

UNIX ist Commandline, basta!

- Umfeldbedingt: Angst und Abscheu (`cmd.exe` ist wirklich nicht zu verwenden)
- Übung macht den Meister
- $\rightarrow$ wir haben bis zum Umfallen geübt

Was bisher geschah: Filesystem

FASCHINGBAUER

Filesystem wird großgeschrieben

- Inhärentes Konzept in UNIX
- Permissions
- Typen: Files, Directories, Links, Device Special Files, ...
- Eine Fülle von Tools

Was bisher geschah: Tools

FASCHINGBAUER

Kleine, feine Tools

- Eine Fülle von Tools, von denen keines zuviel kann
- Kombinierbar über Pipes
- Input und Ouput über Standard-Filedeskriptoren `stdin`, `stdout` und `stderr`
- IO-Redirection mit abenteuerlicher Gestalt und Semantik

Was bisher geschah: Prozesse

FASCHINGBAUER

Prozesse und Threads

- Inhärentes Konzept in UNIX
- Präemptives Multitasking
- Scheduling
- Realtime
- Tools

Ausblick (1)

FASCHINGBAUER

- Keines der vorgestellten Tools birgt ein Geheimnis
- `strace` gibt alles schonungslos preis
- Mit genügend Geschick kann man alles selbst programmieren
- ... vorausgesetzt, man hat das System verstanden

Ausblick (2)

FASCHINGBAUER

Es gibt eine Vielzahl von Bereichen, in denen es Spass macht, zu programmieren. Zum Beispiel:

- File-IO
- Prozesse
- Multithreading
- Netzwerkprogrammierung
- Interprozesskommunikation
- Virtuelles Memory

Goodbye

FASCHINGBAUER

Danke fürs Mitmachen, und vor allem ...

Viel Spass mit dem System!!