Prozeßkontrolle

Komando	Bedeutung
`Ctrl-C`	**Kommando / Programm abbrechen**
`Ctrl-Z`	**Kommandoanhalten**
`fg`	**Kommando weiterlaufen lassen**
`bg`	**Gestopptes Kommando/Programm im Hintergrund weiterlaufen lassen**
`Ctrl-S / Ctrl-Q`	**Ausgabe des Programms anhalten / weiterlaufen lassen**

Shells: Was passiert mit einem Kommando?

- Shell durchläuft folgenden Zyklus:
  - Kommando wird aus der Eingabezeile oder einem Script gelesen
  - Aliases werden expandiert
  - Variable Substitutions werden vorgenommen
  - Wildcards werden expandiert
  - Das Kommando wird ausgeführt:
    - Entweder von der Shell selbst (built-in commands)
    - Oder in einem neuen Prozeß
- Im folgenden Annahme tcsh (csh)

Aliases

- Ersatz für häufige / längliche Kommandos
- Beispiel:

```
% alias mo less
% alias uni "ssh -l zach -X hera.uni-bonn.de
% mo program.cpp
% uni
```

- `alias` zeigt alle Aliases an

Alias 'uni' wird ersetzt ...
Alias 'mo' wird ersetzt ...
Definiert Alias 'uni'
Definiert Alias 'mo'

Variablen

- Variable = Name + Wert, Wert = Zeichenkette
- 2 Arten:
  - Normale Shell-Variablen
  - Environment-Variablen,
    sind auch in Kind-Prozessen (child process) bekannt
- Setzen:

```
% setenv TMP "/tmp"
% set tmpdir = "/home/stud/zach/tmp"
```

- Verwenden:

```
% cp file ${tmpdir}
% echo ${TMP}
```

- Anhängen:

```
% setenv PATH ${PATH}:${HOME}/bin
% setenv PATH ${HOME}/bin:${PATH}
```


Wichtige Environment-Variablen

Variable	Bedeutung
DISPLAY	**Display, auf dem neue Fenster geöffnet werden** (Bsp.: aurikel:0.0)
HOME	**Home-Verzeichnis**
PRINTER	**Default-Drucker**
TMP	**Verzeichnis für temporäre Files**
PATH	**Suchpfad für Kommandos**
MANPATH	**Suchpfad für Man-Pages**
PWD	**Aktuelles Verzeichnis, in dem man sich gerade befindet (CWD)**

- **`printenv`** druckt alle Environment-Variablen aus
- **`echo $VAR`** druckt Wert dieser Environment-Variablen

File Patterns

- File-Name mit Wildcards: `*` `?` `[]`

Wildcard	Bedeutung
`?`	Genau ein beliebiges Zeichen
`*`	Beliebig viele beliebige Zeichen (auch 0)
`{0,1,2}`	Genau ein Zeichen aus der Menge {0,1,2}
`[0-9]`	Genau ein Zeichen aus der Menge {0,...,9}
`[a-zA-Z0-9]`	Genau ein Zeichen aus der Menge {a,...,z,A,...,Z,0,...,9}
`[^0-9]`	Genau ein Zeichen *nicht* aus der Menge {0,...,9}

- Beispiele:

```
% ls *.cpp *.h
```

```
% ls [0-9][0-9]*.ppt
```

```
% ls *[^a-zA-Z0-9_.,-]*
```


Wie die Shell ein Kommando ausführt

- Built-in: Shell führt Kommando selbst aus
  - Beispiel: **`echo`**
- Sonst: externes Programm
  - Beispiel: **`ls dir`**
- **`command`** in **`PATH`** (Environment-Variable) suchen
- Falls nicht gefunden, Fehlermeldung
- Kind-Prozeß erzeugen
  - Erinnerung: erbt Environment des Vater-Prozesses (Shell)
- Argumente (Zeichenketten) dem Prozeß bereitstellen
  - Argumente werden durch Space (i.a.) getrennt
- Warten bis Kind-Prozeß beendet

Suchpfade

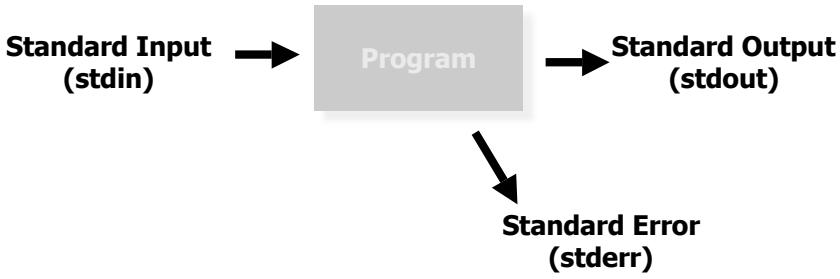
- Environment-Variable mit Liste von durch **:** getrennten Verzeichnissen
- Beispiel:

```
% echo $PATH
.:/usr/bin:/bin:/usr/local/bin:/home/II/zach
```

- Kommandos werden in **PATH** gesucht
  - File mit Namen des Kommandos im ersten Verzeichnis und executable? → ausführen
  - Sonst: nächstes Verzeichnis in PATH untersuchen …
- Analog für Man-Pages und andere

Standard-I/O

- Shell etabliert 3 Kanäle zu / vom Prozeß:



- Default-mäßig mit Terminal (Fenster & Keyboard) verbunden

Redirection

- Verbindet Standard-I/O mit Files
- Prozeß (Programm) bemerkt davon nichts!

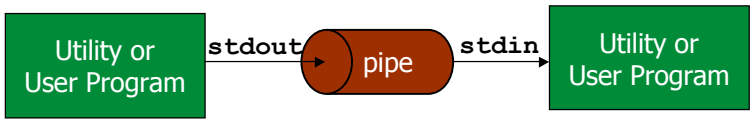
Redirection	Bedeutung
`> file`	**stdout wird nach file geschrieben**
`>> file`	**stdout wird an file angehängt**
`>& file`	`stdout` **und** `stderr` **umlenken**
`< file`	**Programm liest aus file, nicht von Keyboard**
…	**Shell bietet noch viele weitere Möglichkeiten**

- Beispiel:

```
% ls -l > dir-listing
```


Pipelines

- Selber Mechanismus, um Prozesse miteinander zu verbinden:



- Syntax: `command | command | command | …`
- Beispiele:

```
% ls -1 *.cpp | wc -l
% ls -1 *.cpp | sort > sorted-dir
```

Initialisierungs-Files

- Für tcsh: **`~/.tcshrc`** (und **`~/.cshrc`**)
- Inhalt: beliebige Kommandos (shell-built-in & extern)
- Ausgeführt von jeder Shell des Users
- Typischerweise:
  - Environment-Variablen setzen (**`PATH`**, **`MANPATH`**, …)
  - Aliases definieren
  - Completion-Regeln (für Tab an verschiedenen Positionen)
- Achtung: neue Shell aufmachen, falls Änderung
- Empfehlung: Dot-Files von der Vorlesungs-Home-Page installieren

Deutsches UNIX

- Große Unsitte
  - Wegen Terminologie
- **`setenv LANG=en`**

Wichtige Utilities und Programme

- Moving Around
- File-Manipulation
- Filter (oft in Pipelines oder mit Redirection)
- Tools
- Prozeß-Manipulation
- Suchen (find & grep)
- Dokumentation anzeigen

Moving Around

Utility	Funktion
`cd dir`	**Ins Verzeichnis `dir` wechseln (rel. oder abs. Pfad)**
`cd -`	**Ins vorige Verzeichnis zurück wechseln**
`cd`	**Ins Home wechseln**
`pwd`	**Aktuelles Verzeichnis (current working directory) anzeigen**

File- und Verzeichnis-Manipulation

Kommando	Funktion
`rm file`	**File löschen**
`ls [dir]`	**Verzeichnis / File anzeigen**
`ls -l [dir]`	**Mehr Infos zum Verzeichnis / File anzeigen**
`ls -a [dir]`	**Dot-Files (.*) anzeigen**
`cp file1 … dir`	**Files kopieren**
`cp file1 file2`	**Kopie von File1 erzeugen und File2 nennen**
`ln -s file1 file2`	**Symbolischen Link von File2 nach File1 erzeugen (Eselsbrücke: `ln -s` statt `cp`)**
`mv file1 … dir`	**Files verschieben**
`mv file1 file2`	**File umbenennen**
`cat file1 file2 … > file`	**Files aneinanderhängen (konkatenieren)**
`chmod new-perm file`	**Permissions von File ändern**
`mkdir dir`	**Neues Verzeichnis erzeugen**
`rmdir dir`	**Verzeichnis löschen (muß leer sein)**

- Achtung: ES GIBT KEIN RECYCLE-BIN!!! …

Kleine Warnung zu `rm`

Task: Shoot Yourself in The Foot

The proliferation of modern programming languages (all of which seem to have stolen countless features from one another) sometimes makes it difficult to remember what language you're currently using. This handy reference is offered as a public service to help programmers who find themselves in such a dilemma.

```
% ls foot.c foot.h foot.o toe.c toe.o
% rm * .o
rm: .o no such file or directory
% ls
%
```

Permissions modifizieren

- Syntax von **`chmod`** ("change mode"):

  `chmod` *<level><op><perm> filename*

  level = String aus: u, g, o, a (user, group, other, all)
  op = ein Zeichen aus +, -, = (gets, loses, equals)
  perm = String aus: r, w, x, … (read, write, execute, …)

- Beispiele:

```
% chmod u+x foobar
% chmod u+rwx,go-w foobar
% chmod g=u temp/
% chmod u=rwx,g=rwx,o= shared/
```


Tools

Utility	Funktion
`more`	**File anzeigen (`more` ist ein sog. Pager)**
`less`	**Noch besser als `more`**
`head / tail`	**Anfang / Ende des Files ausgeben**
`cat file`	**File ausgeben (keine Funktionalität)**
`echo string(s)`	**String(s) auf stdout (typ. Terminalfenster) ausgeben**
`diff file1 file2`	**Unterschiede zwischen 2 Files anzeigen**
`du -sk dirs …`	**Speicherbedarf der Verzeichnisse in kB anzeigen**
`df -h dir`	**Größe und freien Platz auf einer Platte anzeigen**
`df -hl`	**Größe und freien Platz aller lokalen Platten anzeigen**
`quota -v`	**Freie Quota anzeigen**
`lpr [-Pdrucker] file.ps`	**Postscript-File ausdrucken**
`lpq [-Pdrucker]`	**Printer-Queue anzeigen**
`a2ps [-Pdrucker] file`	**ASCII-File (z.B. Listing) ausdrucken**
`ssh machine`	**Remote auf einem anderen Rechner einloggen**

Utility	Funktion
`id`	Eigene IDs ausgeben
`who / w`	Wer ist eingeloggt?
`date`	Datum anzeigen
`cal`	Jahreskalender anzeigen
`locate file`	Ort von File anzeigen (auch Teilstrings)
`where` *`command`*	Ort(e) von Command anzeigen
`tar czf archive.tgz dirs …`	Komplettes Verzeichnis (inkl. Unterverzeichnisse) zusammenpacken und komprimieren
`tar xzf archive.tgz`	Archiv wieder auspacken
`gzip/gunzip`	File komprimieren / dekomprimieren
`mount /mnt/floppy`	Floppy mounten / unmounten

Prozeßkontrolle

Befehl	Funktion
`ps`	Prozesse anzeigen
`ps -edfjw`	Alle Prozesse anzeigen
`kill pid`	Prozeß mit PID `pid` abbrechen (wie Ctrl-C)
`kill -9 pid`	… wenn der Prozeß trotzdem nicht aufhören will ☺
`command … &`	Prozeß im Hintergrund starten
`nohup`	…
`jobs`	Prozesse im Hintergrund anzeigen
`Ctrl-C`	Foreground-Prozeß abbrechen (interrupt)
`Ctrl-Z`	Foreground-Prozeß anhalten (stoppen)
`fg`	Zuletzt angehaltenen Prozeß im Foreground weiterlaufen lassen
`bg`	Angehaltenen Prozeß im Background weiterlaufen lassen
`Ctrl-S`	Ausgabe des Foreground-Prozesses anhalten (Pr. läuft weiter!)
`Ctrl-Q`	Ausgabe weiterlaufen lassen
`top`	tabellarische Ansicht aller Prozesse und deren CPU-Verbrauch

Filter

- Lesen von stdin, schreiben auf stdout
- Arbeiten oft zeilenweise

Utility	Funktion
`cut`	**Felder oder Zeichenspalten ausschneiden**
`fmt`	**Auf 72 Zeichen umformatieren**
`sort`	**Zeilenweise sortieren (auch nach Teil-Key)**
`uniq`	**Duplikate entfernen**
`wc`	**Zeichen, Wörter, und Zeilen zählen**
`tr`	**Zeichen ersetzen**
`grep`	**Zeichenketten in der Eingabe suchen (s.u.)**
`rev`	**Reihenfolge der Zeichen umkehren (zeilenweise)**

Unix-Philosophie

- "Small is beautiful"
- Make each tool do *one* thing only
- Make it do it well
- Read from stdin, write to stdout (if sensible)
- Use ASCII-Files

Editoren

- Programmierer schreiben ASCII, insbesondere Software
- Heiliger Krieg, welches der beste ist
- Ein Programmier-Editor sollte …
  - Effizientes UI haben (nicht intuitiv!)
    - Wenige Tasten / Mauskilometer für die häufigen Aktionen
  - Syntax Highlighting
  - Makros
  - Reguläre Ausdrücke zum Suchen und Ersetzen
  - Multi-Plattform sein

- Einige Editoren zur Auswahl:
  - vim / gvim (Obermenge von vi, welcher immer installiert ist)
    - Effizientester Editor, steilste Lernkurve
    - Die Homepage von vim: www.vim.org
  - emacs/xemacs (extrem umfangreich)
    - "Emacs wäre gar kein so schlechtes Betriebssystem, wenn es nur einen brauchbaren Editor hätte" ☺
  - nedit (kein non-GUI-Mode)
  - …
- Suchen Sie sich einen hinreichend mächtigen Editor aus, und lernen Sie diesen gut und möglichst vollständig beherrschen!
- Reference-Cards für VIM und Emacs auf der Homepage der Vorlesung

Suchen & Finden

- **`find`**: findet Files aus einem Verzeichnisbaum
  - Kriterien: Name, Datum, Größe, …
  - Aktionen: Filename ausgeben, Löschen, …
- **`grep`**: findet Zeilen in einem File, in denen bestimmte Zeichenketten vorkommen
- **`locate`**: findet Files aufgrund autom. erzeugter DB

Find

- Syntax:

```
find dirs … criteria actions
```

- Kriterien:

```
-name file-pattern
-type file-type
-size file-size
-date …
```

Aktionen:

```
-print
-exec command {} \;
-ls
…
```

Beispiele

- Den File foo im Home finden:

```
% find $HOME -name foo
```

- Wenn man den Namen nicht mehr genau kennt:

```
% find $HOME -iname '*foo*'
```

- Alle JPEG's finden und File-Namen in File schreiben:

```
% find $HOME -name '*.jpg' > image-list
```

- Alle JPEG's größer als 100kB finden:

```
% find $HOME -name '*.jpg' -size +100k
```

- Alle core's löschen:

```
% find . -name core -print -exec rm {} \;
```


Grep

- Syntax:

  grep 'reg-exp' files …

- Varianten: fgrep, egrep
- reg-exp = regular expression (eine Art Pattern-Matching)
- Default: Zeilen ausgeben, die matchen
- Einige Optionen:

  -v Invertierung: Zeilen ausgeben, die nicht matchen
  -i case-insensitive
  -n Zeilennummern ausgeben
  -H Filenamen zu den Matches ausgeben
  -e RE weitere reg-exp's (Oder-Verknüpfung)

Reguläre Ausdrücke

- Besteht aus normalen Zeichen und Meta-Zeichen:
  - Meta-Zeichen: . ? [ ] * + $ ^ \ ( )
  - Normale Zeichen: alle übrigen
- Regulärer Ausdruck = Zeichenkette aus normalen Zeichen und Meta-Zeichen
- Matching:
  - Vergleicht gegebene Zeichenkette und RE von links nach rechts
  - Arbeitet Zeichen ab, falls sie, gemäß Regeln, "übereinstimmen"
  - Arbeitet "greedy"
- Extended RE's

Zeichen	Bedeutung / Match
a	matcht das Zeichen selbst
. (Punkt)	matcht ein beliebiges Zeichen
[abc-f]	matcht ein Zeichen aus {a,b,c,d,e,f}
[^abc]	matcht ein Zeichen *nicht* aus {a,b,c}
^ $	stehen für den Anfang/Ende der Zeile
a?	a ist optional ("schluckt" a, falls vorhanden)
a+	a muß einmal oder öfter vorkommen
a*	a darf belieb oft, auch keinmal, vorkommen
(*RE*)	Gruppierung
RE1|*RE2*	matcht a oder b
\	hebt Bedeutung des nachfolgenden Meta-Zeichens auf
...	

Beispiele

- **`grep 'abc' file`**:
  - alle Zeilen, die "abc" enthalten
- **`grep 'a.c' file`**:
  - alle Zeilen, die "axc" enthalten, wobei x beliebiges Zeichen ist
- **`grep -e 'a[x-z]b' -e 'c[u-w]d' file`**:
  - "axb" oder "cud" oder "ayb" …
- **`grep 'a\[[^]]*\]=' file`**:
  - alle Vorkommen der Form "a[…]=", wobei … eine beliebig lange Zeichenkette ist, die kein] enthält
- **`grep 'a\[[^]]*\] *=' file`**:
  - wie vorher, mit beliebig vielen Spaces zwischen "]="

More input? (Hilfe)

- 4 Arten von Informationsquellen:
  - Man Pages
  - Info pages
  - HTML-Seiten
  - Unter KDE: Start → Help, dann z.B. "UNIX manual pages"
- Man Pages:
  - 'man cmd' – Man-Page zu cmd anzeigen (Programm oder Funktion)
  - man –k keyword – Alle Man-Pages nach keyword durchsuchen (nur die 1-zeiligen "Köpfe" jeder Seiten)
  - man –K keyword – Alle Man-Pages nach keyword durchsuchen (komplette Seite)
  - Start-Menü → Help

Format of each man page

Name	Name und 1-zeilige Beschreibung
Syntax	
Description	Ausführliche Beschreibung
Options	
Files	Liste von Files wichtig für diesen Befehl
Return values	
Diagnostics	Mögliche Fehlermeldungen und Ursachen
Bugs	Bekannte Bugs und Unzulänglichkeiten
See also	Verwandte Befehle und Infos

Info & HTML

- Info-Seiten:
  - 'info cmd' – Sog. Info-Seite in einem einfachen Viewer anzeigen
  - Hierarchisch (angelehnt an HTML)
  - Steuerung:
    - **`Return`** auf Stern – In Unterseite springen
    - **`u`** – zur übergeordneten Seite zurück
    - `/string` – suchen
- HTML-Seiten:
  - Hauptproblem: finden
  - Normalerweise in **`/usr/share/docs`**
  - Hilfsmittel: **`locate`**

Minimalwissen

- Folgende Befehle sollten Sie gut beherrschen:
  - ls, cd, mkdir, rmdir, cp, mv, rm, ln, chmod,
  - less, cat, sort, wc
  - grep, find
  - ps, jobs, kill
  - Später: make, g++, gdb
- Lesen Sie deren Man-Pages (immer wieder)

Grundregeln unter UNIX

- Don't Panic!
- RTFM! ("read the f*ing manual")
- Probieren geht über studieren …

Weiterführender Kurs

- Aus Göttingen über ELAN importiert
- Übertragung dienstags per Videokonferenztechnik in den Diplomandenraum des IEE
- Siehe:
  - http://www.tu-clausthal.de/odin/scripts//DisplayVorlesung.cgi?sem=WS2005&id=W+8824&s=de
  - http://www.uni-math.gwdg.de/linuxuebung/