

Programmierwerkzeuge unter Unix

Tutorium der Unix-AG

Jean-Marie Gaillourdet

Christian Schmidt

Mathias Dalheimer

www.unix-ag.uni-kl.de

Keine IDE am Anfang benutzen! Stattdessen:

- Die Shell als Grundlage
- Editoren
- Build-Tools: Make
- Ausblick

Wie kann ich mir selber helfen?

Voraussetzung: Wissen aus der
Rechnereinführung

- Die Shell ist eine eigene Programmiersprache
- Unix-Prinzip: Viele kleine Programme
- Shell verknüpft diese (Pipes, Ausgabeumleitung)
- Wichtig: Konzept der Umgebungsvariablen

```
[hannibal@zoidberg hannibal]$ env | grep HOME  
FORREST_HOME=/usr/java/forrest  
JAVA_HOME=/usr/java/j2sdk  
HOME=/home/hannibal
```

Umgebungsvariablen

- Umgebungsvariablen sind Shell-spezifisch
- Um diese global verfügbar zu machen, muß man sie exportieren (hier: csh)

```
m_dalhei@domino[~] setenv TEST md  
m_dalhei@domino[~] echo $TEST  
md
```

- Andere Kommandos bei der bash:

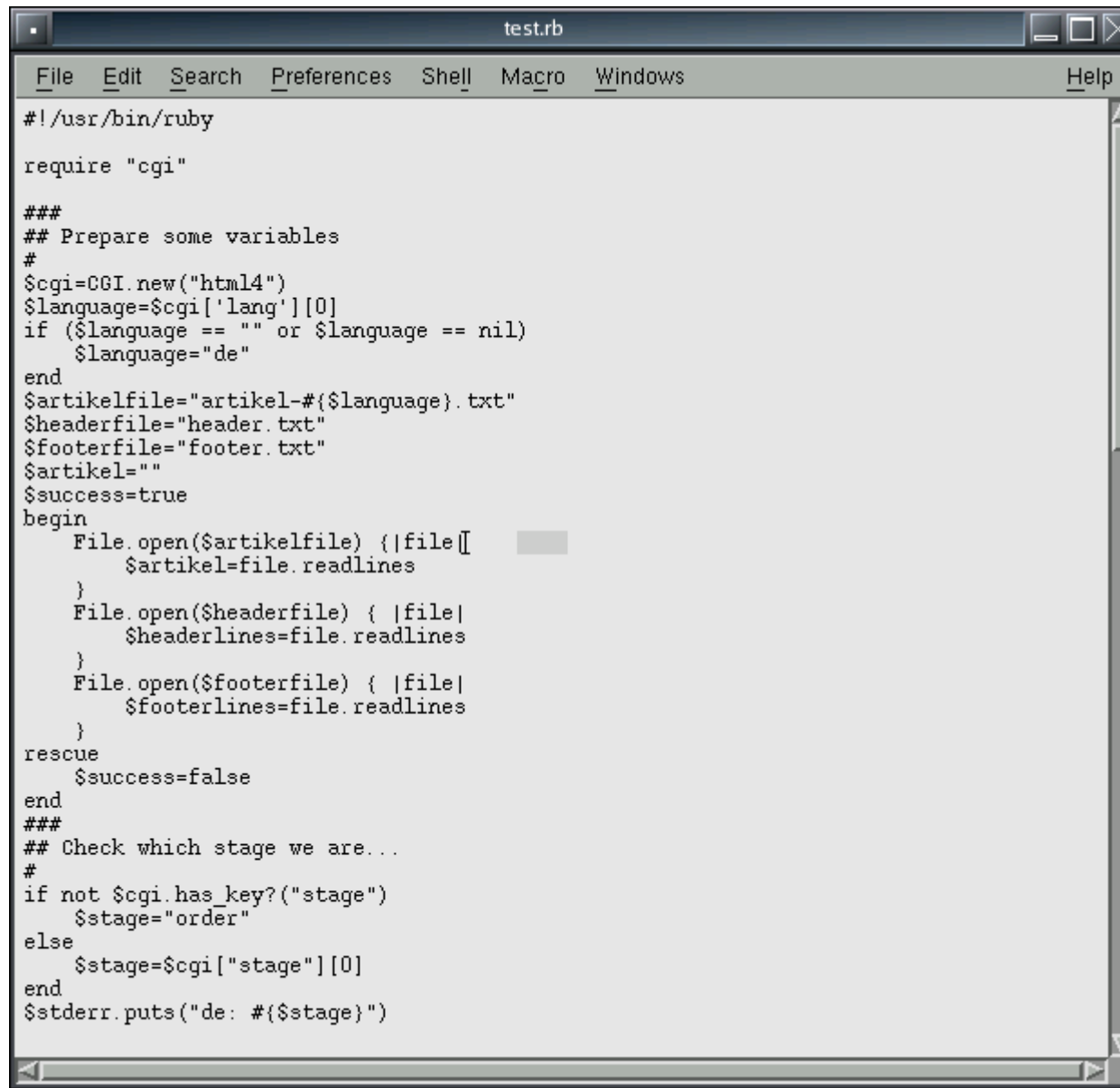
```
[hannibal@zoidberg hannibal]$ TEST=md  
[hannibal@zoidberg hannibal]$ export TEST  
[hannibal@zoidberg hannibal]$ echo $TEST  
md
```

Wichtige Umgebungsvariablen

- Viele Funktionen werden über Umgebungsvariablen gesteuert:
 - \$HOME: Das Home-Verzeichnis
 - \$PWD: Das aktuelle Verzeichnis
 - \$CLASSPATH: Die Includes für Java
- Die Umgebungsvariablen sind flüchtig, müssen also bei jedem Einloggen gesetzt werden.
- Dafür: Profildateien wie .cshrc

- Loggt Euch bitte ein und öffnet ein Terminalfenster.
- Schaut Euch eure Umgebungsvariablen an.
- Setzt die Umgebungsvariable UNIX auf den Wert AG.
- Erzeugt ein Arbeitsverzeichnis namens „tutorium“. Wechselt da hinein!
- Spielt mit dem Verzeichniswechseln und dem Befehl ls herum. Welche Optionen hat der ls-Befehl?

- Dienen zur Verarbeitung von Texten
- Guter Programmiereditor muß folgende Features haben:
 - Komfortable Bedienung (verschiedene Konzepte)
 - Syntax-Highlightning (mehrere Sprachen)
 - Umfassende Konfigurationsmöglichkeiten, z.B. Tabstops
 - Mehrfaches Undo, ..
- Später noch genaueres zu: nedit, vim, emacs



```
test.rb
File Edit Search Preferences Shell Macro Windows Help
#!/usr/bin/ruby
require "cgi"

###
## Prepare some variables
#
$cgi=CGI.new("html4")
$language=$cgi['lang'][0]
if ($language == "" or $language == nil)
  $language="de"
end
$artikelfile="artikel-#{ $language }.txt"
$headerfile="header.txt"
$footerfile="footer.txt"
$artikel=""
$success=true
begin
  File.open($artikelfile) { |file|
    $artikel=file.readlines
  }
  File.open($headerfile) { |file|
    $headerlines=file.readlines
  }
  File.open($footerfile) { |file|
    $footerlines=file.readlines
  }
rescue
  $success=false
end
###
## Check which stage we are...
#
if not $cgi.has_key?("stage")
  $stage="order"
else
  $stage=$cgi["stage"][0]
end
$stderr.puts("de: #{ $stage }")
```

Heute werden wir diesen einfachen Editor benutzen. Starten mit "nedit"

- Starte und öffne damit die Datei `$HOME/.cshrc`
- Schau Dir an, was darin steht
- Definiere dort eine Umgebungsvariable `$CLASSPATH` mit dem Wert „test“
- Prüfe, ob Deine Änderungen auch wirklich in alle Shells übernommen werden: Dazu öffne weitere Terminals und schau Dir dort die Umgebung an!

Shellprogrammierung

- Die Shell ist nicht nur ein interaktives Werkzeug!
- Zum Automatisieren kleinerer Aufgaben ist die bash das Werkzeug der Wahl!
- Die Programme werden einfach in Textdateien hinterlegt
- Die erste Zeile muß auf den Interpreter zeigen:
`#!/bin/bash` (oder ähnlich)
- Die Textdatei muß ausführbar gemacht werden:
`chmod u+x <datei>`
- Dann: Starten des Skripts durch den Dateinamen

- Zehnmal hallo sagen:

```
#!/bin/sh
```

```
echo "starting...."
```

```
for i in 1 2 3 4 5 6 7 8 9 10 ;
```

```
do
```

```
    echo "Hallo $i"
```

```
done
```

- echo gibt etwas auf dem Bildschirm aus
- for ist eine Schleife, i die Schleifenvariable

Abfragen von Parametern

- Die Shell kann auch mit Parametern umgehen:

```
#!/bin/sh
```

```
echo "Parameter 1 lautet: $1"
```

```
echo "Alle Parameter sind: $*"
```

```
echo "Meine PID ist: $$"
```

- Eine Ausgabe kann so aussehen:

```
randy@zoidberg:~/prog_unix-2> ./example2.sh hallo hiuew
```

```
Parameter 1 lautet: hallo
```

```
Alle Parameter sind: hallo hiuew
```

```
Meine PID ist: 25789
```

- Schreibe ein kleines Bashscript, das „Hallo Welt“ auf dem Bildschirm ausgibt und teste es
- Ändere Dein Script so, das Du Deinen Namen als Parameter angeben kannst und die Ausgabe „Hallo <Name>“ lautet
- Überlege, wie man die Variablen vom Anfang des Tutoriums in Dein Script einbauen könnte!

Arbeiten mit Textdateien

- Oft muß man nach einem Text suchen
 - grep stellt hierfür Funktionen zur Verfügung:
grep <regex> Datei
- Beispiel:

```
randy@zoidberg:~/prog_unix-2> echo "Dies ist ein Test" > test.txt
randy@zoidberg:~/prog_unix-2> grep '[e]*n' test.txt
Dies ist ein Test
randy@zoidberg:~/prog_unix-2> grep '[e]n' test.txt
```
- Die manpage hilft weiter
- Reguläre Ausdrücke sind sehr mächtig!

find sucht nach Dateien

- find untersucht nicht den Inhalt, sondern die Metadaten
 - Dateiname, Letzte Änderung, ...
 - Man kann auch Befehle über den Suchergebnissen ausführen
- Beispiel:

```
randy@zoidberg:~/prog_unix-2> find . -name 'ex*[^~]' -exec ls -il {} \;  
148828 -rwxr--r--  1 randy  users   91 2003-10-23 16:00 ./example1.sh  
148836 -rwxr--r--  1 randy  users   96 2003-10-24 11:32 ./example2.sh
```

- Findet alle Dateien in Eurem Homeverzeichnis!
(Tip: man find)
- Welche Ausgabe liefert:
`grep "setenv" ~/.cshrc`
Warum?

- Macht Euch mit den Befehlen vertraut!
- In den nächsten Veranstaltungen geht es um:
 - Thema Entwicklungs-Tools: Mi., 05. November
 - Thema Editoren: Mi., 12. November
 - Jeweils 15:30 in 48/231
- Bei Fragen:
 - In der Unix-AG vorbeikommen (34/116)
 - mathias@unix-ag.uni-kl.de