Weiterführendes Linux-Tutorium

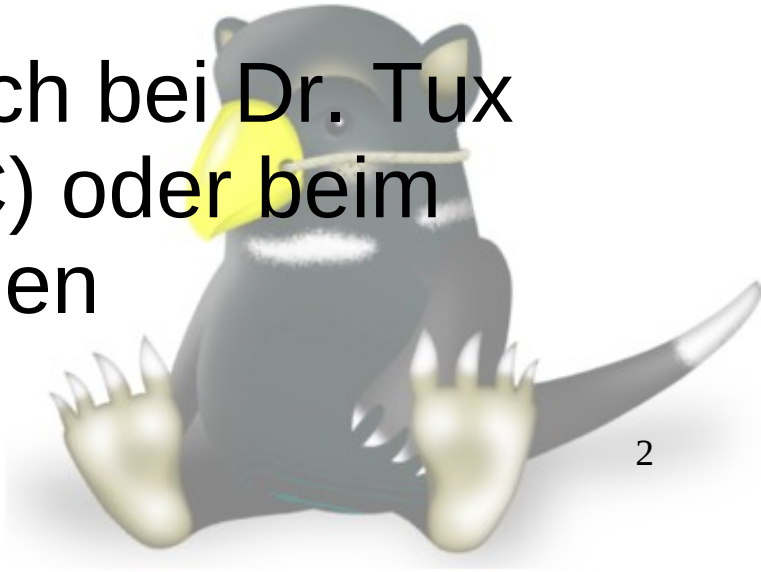
Willkommen beim weiterführenden Linux Tutorium.

Veranstalter & Kontakt

- Daniel Clemens
  (d.clemens@stud.fh-sm.de)

- Christopher Ezell
  (c.ezell@stud.fh-sm.de)

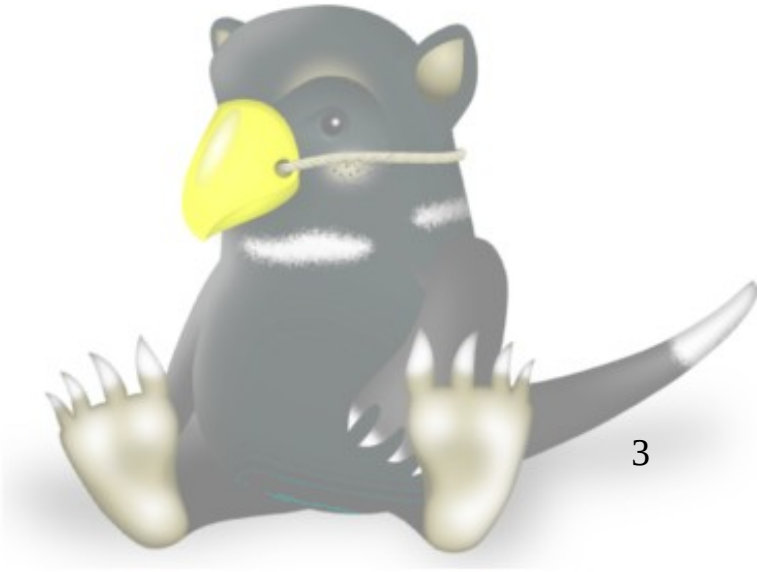
- Allgemeine Fragen können auch bei Dr. Tux (Mittwochs 15-17Uhr im DBCC) oder beim Linux Stammtisch geklärt werden



Termine

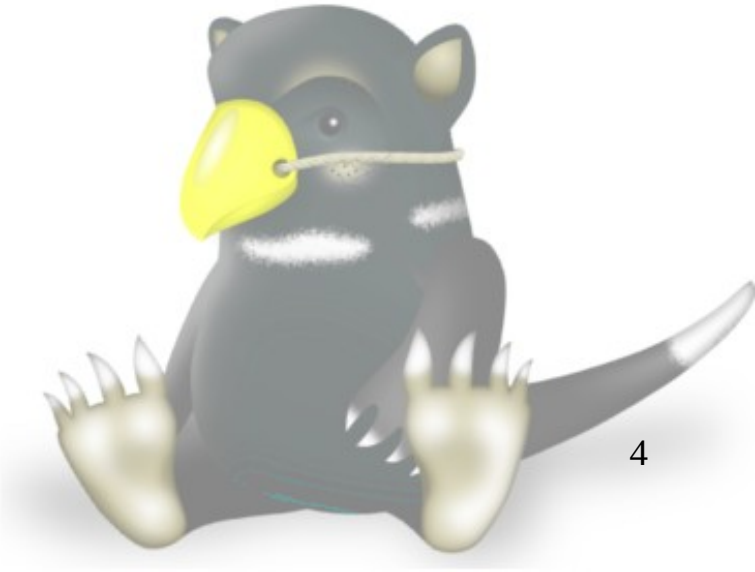
- Wöchentlich Mittwoch ab dem 06. Mai 2009
- 14.15 – 15.45 Uhr im PC Pool 3
- Folien per Download:
  http://lip.fh-schmalkalden.de

  http://linux.rockt.es



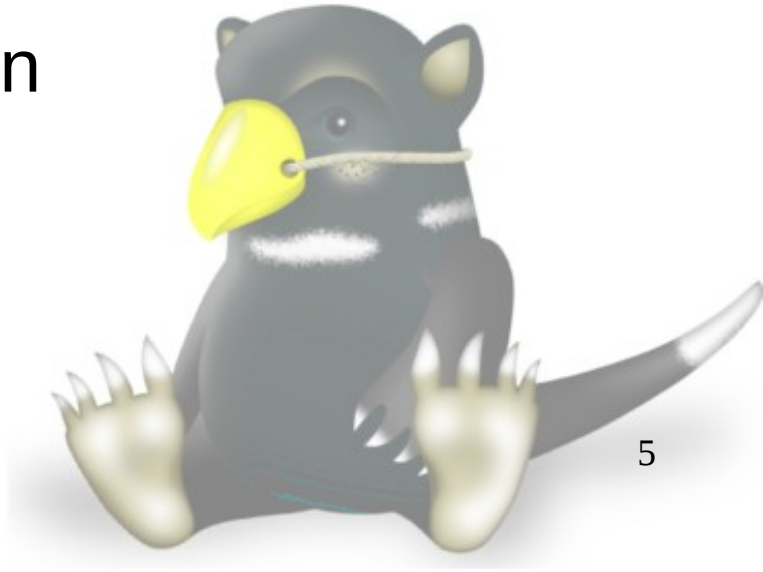
Linux Stammtisch

- jeden 2. Mittwoch im Monat um 19.00 Uhr
- nächstes Treffen 13. Mai im Eiscafe Hellmund gegen über der FHS
- Fragen bezüglich Linux im privaten Einsatz
- Hilfe bei Konfigurationen & speziellen Einstellungen
- alles im gemütlichen Umfeld



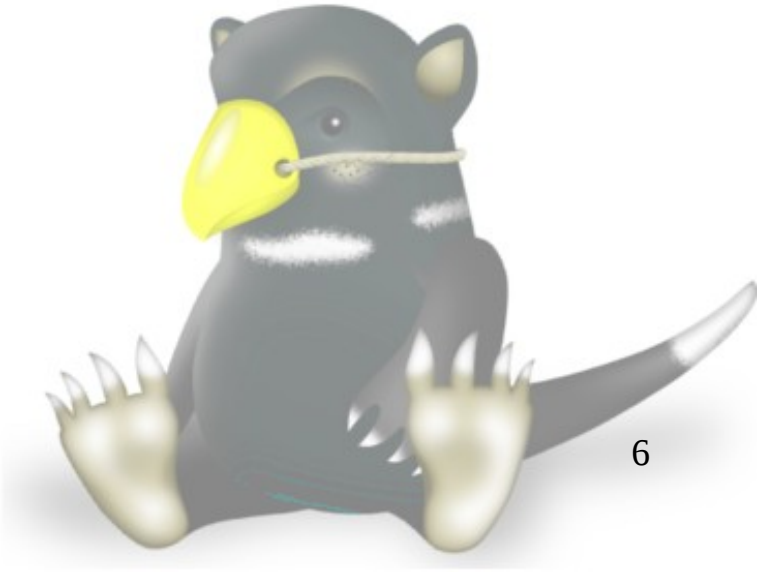
Inhalt – Was wollt ihr machen?

1. Allgemeine Konsolenbefehle/ -skripte
2. Init-Prozess/ Bootloader / Bootconfig-Dateien
3. Der Linux-Kernel
4. Netzwerkkonfiguration und -tools
5. Linux-Server
    1. Installation und Konfiguration
    2. Umgang mit Serverdiensten



Abschluss

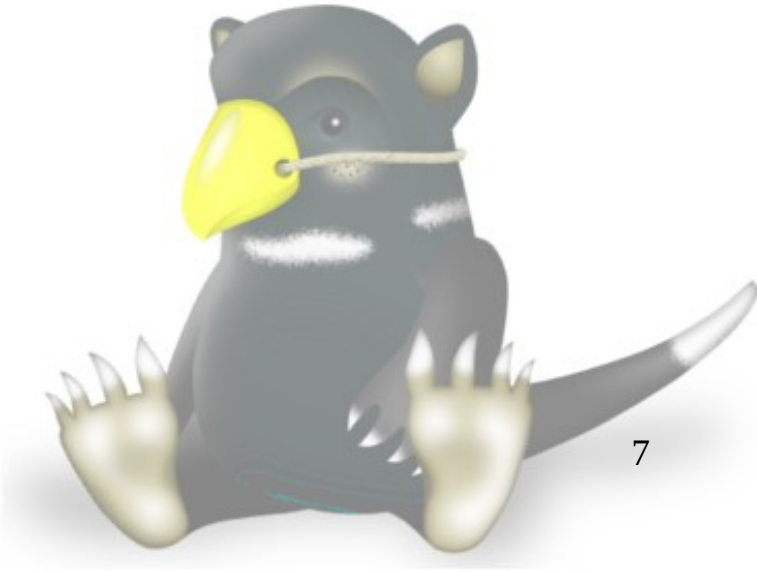
- 45 minütiger Test
- Bei Bestehen Zertifikat
- Prüfung am 17. Juni 2009



Literatur

- Linux – Installation, Konfiguration, Anwendung
  erschienen bei Addison-Wesley, München
  Author: Michael Kofler

- Linux openbook:
  http://openbook.galileocomputing.de/linux/

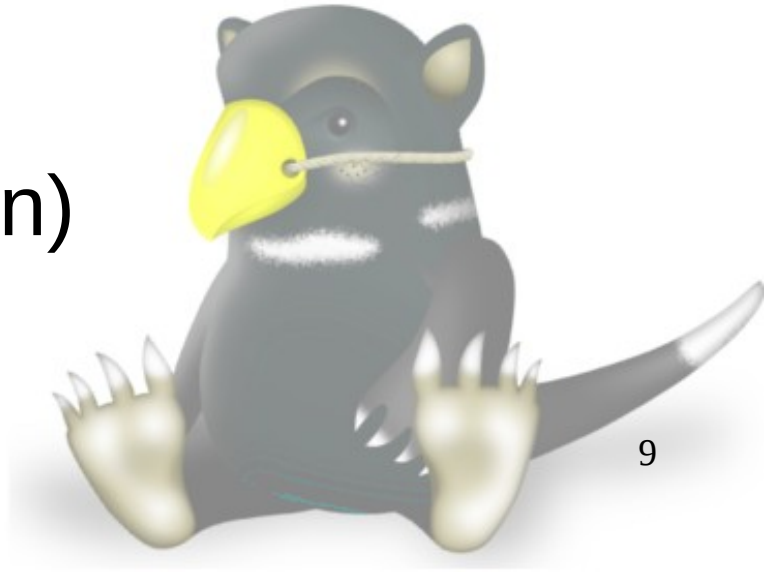


1.Konsole intensiv

- Die Konsole

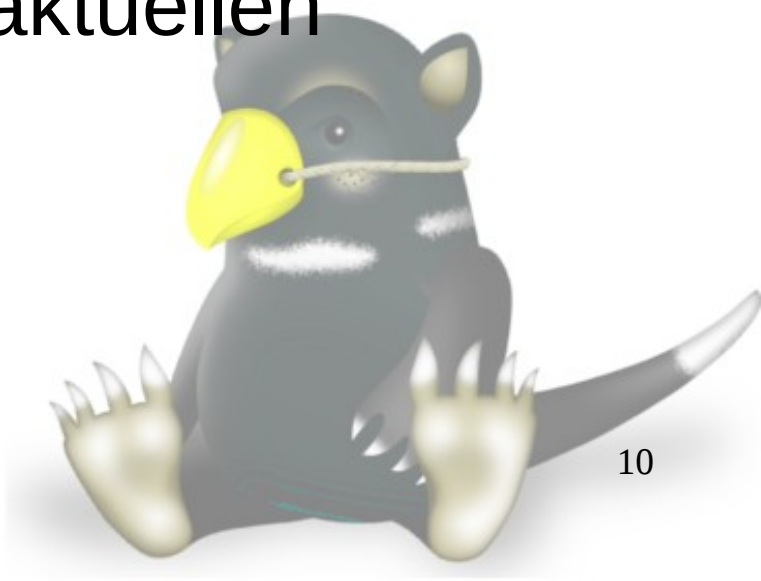
Grundlegende Konsolenbefehle

- ls – Verzeichnisse auflisten
- cd – Verzeichnis wechseln
- mkdir – Verzeichnis erstellen
- mv – Dateien und Verzeichnisse umbenennen
- rm – Dateien und Verzeichnisse löschen
- find – Kontrolliertes auflisten
- grep – Suchen (auch in Dateien)



Spezielle Befehle

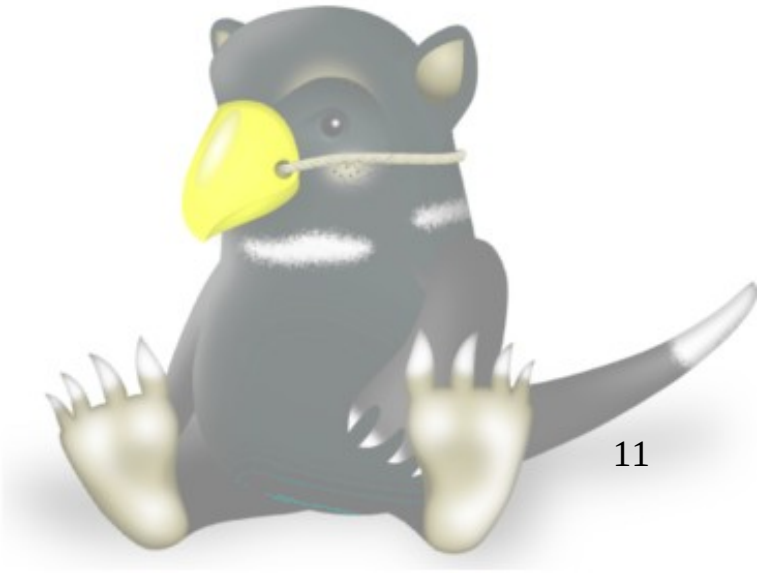
- tee – verdoppelt die Ausgabe des vorangegangenen Befehls
- alias – legt einen Alias von einen Befehl an
  alias more=less
- chmod / chgrp / chown – Dateirechte ändern
- grepall – listet alle Dateien im aktuellen Verzeichnis



Spezielle Techniken

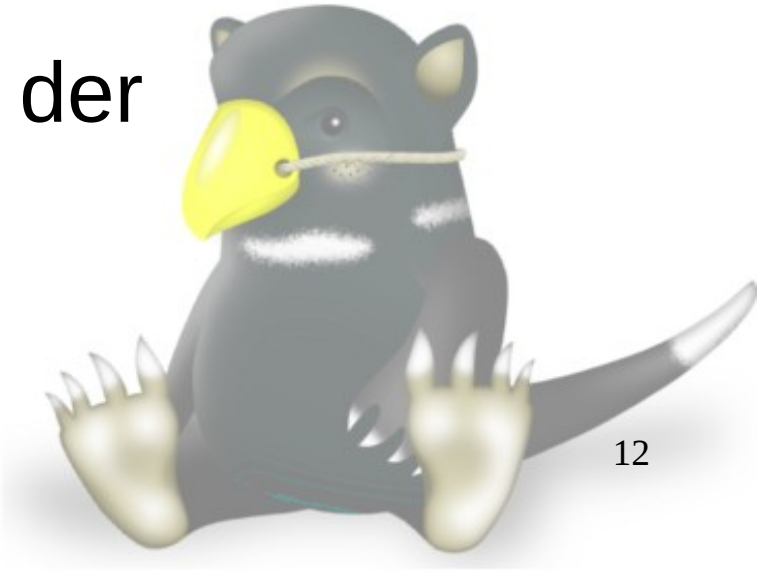
- Ausgabeumleitung
  ls *.tex > inhalt
- Priorisierung der Befehle
  (ls; date) > inhalt
- Bereichsauflistung von Verzeichnissen

  echo /[a-f]*
- Verwenden von Pipes
  ls -l | less



Shell Variablen

- Sind in der Shell über den $-Operator zu erreichen
- Es gibt ein vom System bereitgestellten Satz von Shell-Variablen
- Man kann sich aber jederzeit auch eigene Variablen definieren (var=abc)
- Diese Technik ist vor allem bei der Verwendung von Skripts sehr vorteilhaft

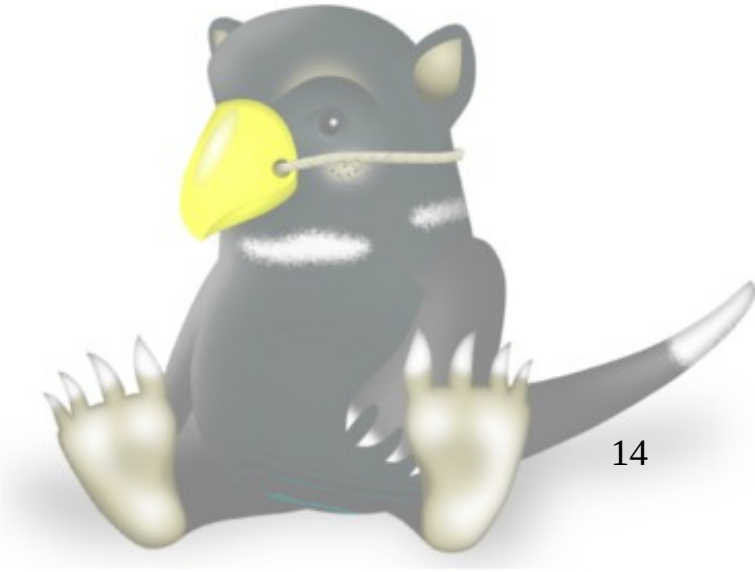


Shell Variablen

- Vordefinierte Varibalen:
  - BASH : Dateinamen der Bash
  - HOME: Pfad des „Home“-Verzeichnisses
  - LOGNAME: Name des aktuellen Users
  - HOSTNAME: enthält den Hostnamen
  - PATH: enthält eine Liste von Verzeichnissen die bei der Eingabe eines Befehls nach diesem durchsucht werden

echo - Befehl (Spezielle Verwendung)

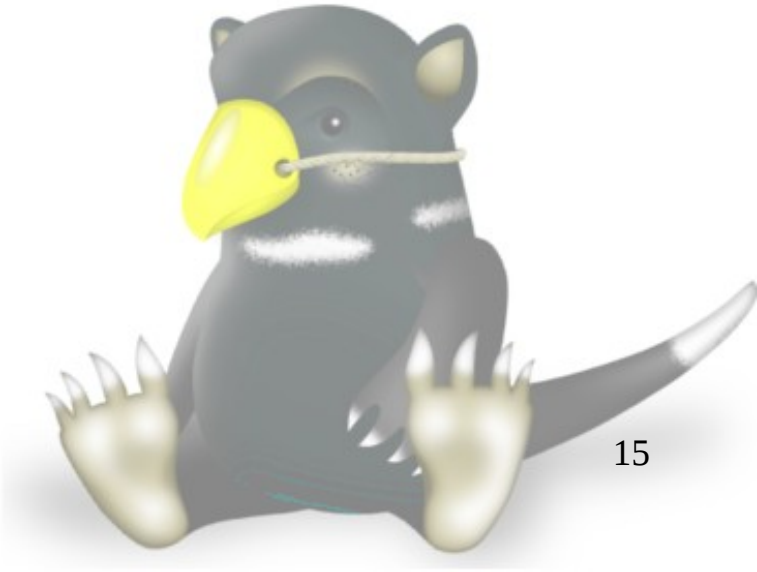
- Einfache Programm API
  echo $[4*9] oder echo $[12>13]
- Bereichsausgabe (Pipeinput)
  echo {1..6}
- Verwendung zur Permutation von Symbolen
  echo {a,b}{1..4}
- Ausgabe von Shellvariablen
- echo „Pfad: $PATH“



Scripting

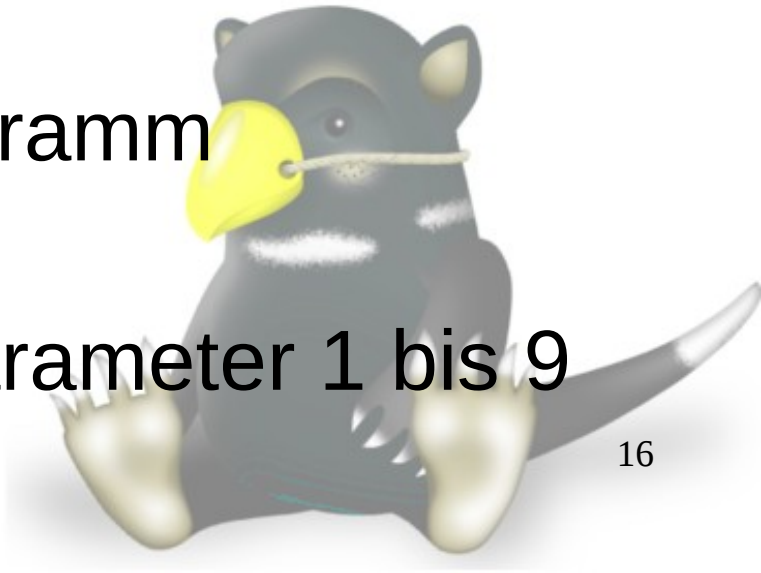
- Einführendes Beispiel:

```
#!/bin/sh
#Verwendung: makethumbs *.jpg
if [ ! -d 400x400]; then #Unterverz. Erzeugen
 mkdir 400x400
fi
for filename do                                    # alle Dateien verarbeiten
 echo „processing $filename“
 convert -size 400x400 -resize 400x400 $filename 400x400/$filename
done
```



$-Variablen

- $? - Rückgabewert des letzten Kommandos
- $! - PID des zuletzt gestarteten Hintergrundprozesses
- $$ - PID der aktuellen Shell
- $0 - Dateiname des gerade laufenden Scripts (oder symbolische Link)
- $# - Anzahl der dem Shellprogramm übergebenen Parameter
- $1 bis 9 - Die übergebenen Parameter 1 bis 9



Kontrollstrukturen - if

```
#! /bin/sh
# Beispiel iftest
if test $# -ne 2; then
 echo „Fehler!“
 exit
else
 echo „Parameter 1: $1, Parameter 2: $2“
fi
```

- Was macht das Programm?
- Wie wird das Programm aufgerufen?
- Wie werden die Parameter übergeben?

Kontrollstrukturen - for

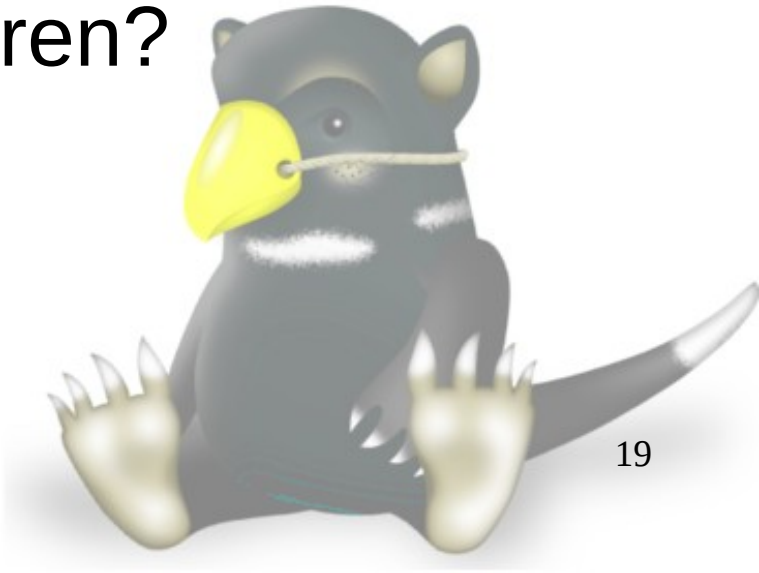
```
#! /bin/sh
for i in a b c; do
 echo $i
done
```

- Wie nennt man diese Kontrollstruktur?
- Was wird ausgegeben?
- Warum braucht man bei einen Shellscript die Endung .sh?

Kontrollstrukturen - while

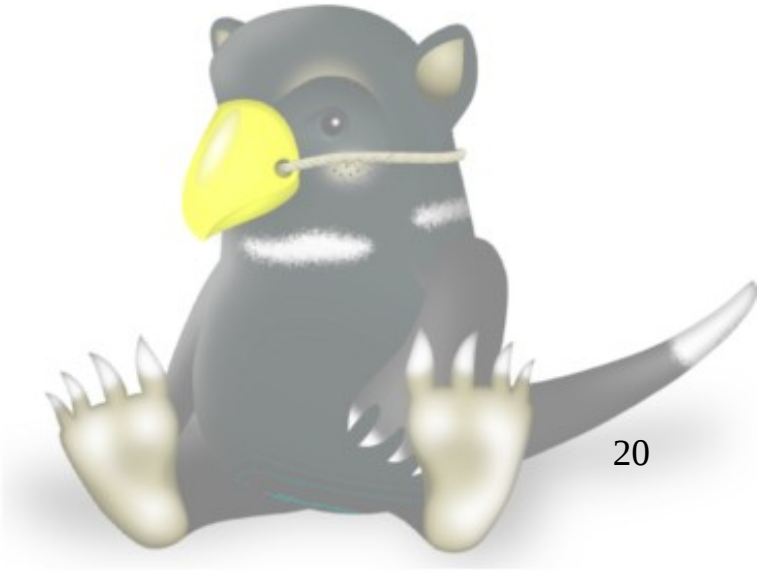
```
#! /bin/sh
i=1
while test  $i -le 5
 do
 echo $i
 i=`expr $i + 1`
done
```

- Was macht das „expr“?
- Was würde ohne „expr“ passieren?



Zusammenfassung

- Wiederholung der Konsolenbefehle
- Spezielle Konsolentechniken
- Shell Variablen
- Vordefinierte Variablen
- Scripting
- Kontrollstrukturen



Fragen?

- Wir stehen noch für Fragen zur Verfügung. Ihr könnt uns auch gerne eine E-Mail schreiben, falls es noch Unklarheiten gibt.
- Die Folien stehen demnächst auf der LIP-Seite (bzw. http://linux.rockt.es) zur Verfügung.