1 Begrüßung

Hallo zusammen. . . ..

2 Betriebssysteme

Ein Betriebssystem dient dazu:

- einen einheitlichen Zugriff auf Hardware zu ermöglichen.
- die Rechenzeit der CPU auf mehrere Prozesse zu verteilen (Multitasking)
- auf Mehrbenutzersystemen zur Rechteverwaltung

2.1 große Betriebssystemfamilien

- DOS
  - CP/M
  - DOS
  - Windows
- UNIX
  - Linux/GNU
  - Solaris
  - Free/Open/Net-BSD
  - Mac OS-X
  - iOS (iPhone)
  - Android

2.2 Start des Betriebssystems

Nachdem einschalten des Computers fährt die CPU die erste Stelle in ihrem Speicher an, und beginnt das dortige Programm abzuarbeiten.

Dort befindet sich ein Festspeicher-ROM das das BIOS[1] enthält. Je nach System heist das BIOS auch mal anders.

Das BIOS enthält grade genug Informationen um auf eine Festplatte, einen USB, zu lesen und auszuführen, oder ein ROM auf einer Netzwerkkarte zu starten das einen Bootlader über das Netzwek holt.

Der Bootlader dient dazu ein Auswahlmenü für verschiedene Betriebssysteme bereitzustellen und läd dann den Betriebssystem-Kernel/Systemkern und führt diesen aus.

Der Systemkern läd jetzt die benötigten Hardware Treiber und startet dann den ersten Prozess den Init Prozess.

Der init prozess führt eine reihe von Skripten aus, konfiguriert das Netzwerk, startet noch einige Systemdienste und schließlich einen oder mehrere Anmelde Prozesse. Und schon ist das Betriebssystem gestartet.

2.3 Multitasking/Multiuser

Linux ist genau wie alle anderen Unix Systeme ein Multitasking/Multiuser Betriebssystem, das Bedeutet:

1. Es können mehrere Programme/Prozesse gleichzeitig ausgeführt werden.
   Das wird durch die Einteilung der Rechenzeit aller Programme in kleine Zeitschlitze, und das sequenzielle abarbeiten dieser Zeitschlitze. Das Scheduling übernimmt dabei das Betriebssystem.
2. Es können mehrere Personen gleichzeitig an einem Rechner arbeiten.
   Dieses feature wurde ursprünglich durch mehrere Datenterminals[2] erreicht die an den Computer angeschlossen waren, heutzutage loggt man sich eher über Netzwerk auf einem anderen Computer ein, oder man wechselt auf der Hauptkonsole zwischen mehreren eingeloggten Benutzern.

[1] Basic Input Output System
[2] Monitor/Tastatur einheiten

3 Zugriff auf den Computer

Die Bedienung eines Computers kann man in drei Kategorien einteilen:

1. Textkonsole/Terminal/Commandline
2. Graphische Benutzeroberflächen
3. Experimentelle 3D Benutzeroberflächen

Wir wollen uns Insbesondere mit der Bedienung über die Commandline auseinandersetzen.

- Monitor/Tastatur
  Natürlich kann man mit Monitor und Tastatur auf einem Computer arbeiten. Linux auf IBM Kompatiblen rechnern bietet 7-8 Virtuelle konsolen zwischen denen man mit $(Alt)+(F1-F8)$ umschalten kann. Falls auf einer Konsole die Grafische Umgebung X11[3] läuft muss $(Ctrl)+(Alt)+(F1-F8)$ für den Konsolenwechsel verwendet werden.
- Login Netzwerk SSH
  Für den Konsolen login über Netzwerk wird heutzutage das Protokoll SSH[4] verwendet. Bis vor 12 Jahren waren auch vereinzelt noch andere unverschlüsselte verfahren anzutreffen.
- serielles Terminal
  Von historischer Bedeutung ist der Login über ein Serielles Terminal, das sind Monitor/Tastatur einheiten die über eine RS232 Schnittstelle an den Computer angeschlossen waren. Der Unterschied eines Terminals zu einem modernen Monitor ist vor allem der das zwischen Terminal und Computer Schriftzeichen übertragen werden ein Moderner Monitor zeigt Bildpunkte/Pixel an.

Übung

- Putty Installieren, Einloggen ...
- Konsolen umschalten und einloggen ...

Wenn alles glatt geht erhalten wir einen Systemprompt der mit $ endet.

```
zem@denkbrett:~$
```

4 Client/Server

Kurzer einschub um die Begriffe Client und Server zu klären.
In der Softwarewelt ist der Client genau wie in der richtigen Welt als bittsteller zu sehen, während der Server (engl. Diener) einen Dienstanbieter darstellt.
Im Fall von unserem SSH login ist das programm Putty der ssh client, also ein Programm das auf dem eigenen Rechner stellt und einen Virtuelle Konsole in einem Fenster darstellt. Dafür muss sich Putty mit einem ssh Server verbinden, der ein Virtuelles Terminal als Dienstleistung bereitstellt.

Ein weiteres Beispiel für Client/Server Architektur ist ein Web Browser, dieser stellt über das dazugehörige Netzwerkprotokoll http [5] eine Anfrage für eine Webseite an einen Web Server der diese dann an den Client übermittelt.

Die Computer auf denen so ein Server läuft unterscheiden sich prinzipiell nicht von denen auf denen ein Client läuft, oft wird jedoch der ganze Computer auf dem der Server läuft als Server bezeichnet.

[3] Typischerweise Konsole 8
[4] Secure Shell
[5] HyperText Transport Protocoll

5 Wie funktioniert eine Komandozeile

Das Konzept der Komandozeile stammt noch aus der Zeit vor der Erfindung des Monitors, damals wurde die Ausgabe einfach auf Papier gedruckt. Zu dieser Zeit gab es auch noch Tastaturen ohne Cursortasten, die auch später noch auf den ersten Terminals mit Monitor eingesetzt wurden. Daraus folgt der Aufbau der Komanndozeile:

Das $ Zeichen oder, wenn es sich um den Administrator (root) handelt, das # zeichen zeigt an das auf eine Eingabe gewartet wird. Hin und wieder werden auch andere Zeichen konfiguriert, ebenso kann eine Zeichenkette vor dem $ zeichen konfiguriert werden, meistens steht hier der Benutzername, der Rechnername sowie das aktuelle Verzeichnis.
Wir schreiben in die Zeile *„echo Hallo Welt“* und bestätigen mit Enter (Zeilenumbruch).

```
zem@denkbrett:~$ echo Hallo Welt
Hallo Welt
zem@denkbrett:~$
```

Wir haben damit unser erstes Komando, *echo*, gelernt und stellen fest das alles was dahinter steht ausgegeben wird. Den Teil hinter dem Komando nennen wir **Parameter**. Diese Parameter werden durch Leerzeichen getrennt. Sollten wir einmal einen Parameter mit Leerzeichen verwenden müssen, so können den Parameter in ẗsetzen oder das Leerzeichen mit einem \ maskieren:

```
zem@denkbrett:~$ echo "Hallo Welt"
Hallo Welt
zem@denkbrett:~$ echo Hallo\ Welt
Hallo Welt
```

6 Filesystem Hierachie Standard

Die meisten Betriebssysteme ordnen ihre dateien in Strukturen aus Verzeichnissen und Unterverzeichnissen, das macht Linux grundsätzlich genau so. Von Windows her mag man es gewohnt sein, das es verschiedene Platten gibt deren Dateisystem durch einen Laufwerksbuchstaben (C:\, D:\) zugeordnet wird, in diesem Punkt unterscheiden sich die beiden Systeme jedoch gewaltig.

Linux hat ein sogenanntes Virtuelles Dateisystem (VFS). Der Trenner zwischen den Verzeichnissen ist der / (slash[6]). Das Dateisystem startet Systemweit mit dem Wurzel oder Rootverzeichnis / Laufwerksbuchstaben wie von Windows gewohnt gibt es nicht.

Die Speichermedien des Computers werden zusammen mit einigen Virtuellen Dateisystemen dynamisch und im Namen frei konfiguriuerbar in Verzeichnisbaum eingehangen (gemountet). Das hat den Vorteil das man die physikalischen Speicherorte für dateien und Programme beliebig verlegen kann ohne allzuviele änderungen am System machen zu müssen.

Das kommando mount ohne parameter zeigt uns an, an welchen stellen im Dateisystem von einem Medium auf ein anderes gewechselt wird.

```
zem@denkbrett:~$ mount
none on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
none on /proc type proc (rw,nosuid,nodev,noexec,relatime)
none on /dev type devtmpfs (rw,relatime,size=1030736k,nr_inodes=220823,mode=755)
none on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620)
/dev/disk/by-uuid/0e1c6352-685a-46d3-9569-9b133a104301 on / type ext4 (rw,relatime,errors=remount-ro,us
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=206516k,mode=755)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev,relatime,size=413032k)
tmpfs on /run/shm type tmpfs (rw,nosuid,nodev,relatime,size=413032k)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
none on /proc/bus/usb type usbfs (rw,relatime)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
```

[6]DOS und Windows trennen mit einem \ (backslash)

```
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,nosuid,nodev,noexec,relatime)
/home/zem/.crypro.img on /home/zem/crypto_fs type ext3 (rw,nosuid,nodev,relatime,errors=continue,barrie
zem@denkbrett:~$ ls /
bin      bzImage.2.6.23  bzImage.2.6.36.2  dev   initrd      media  proc           root  selinux  tmp
boot     bzImage.2.6.26  bzImage.2.6.37    etc   lib         mnt    recovery       run   srv      usr
bzImage  bzImage.2.6.31  bzImage.2.6.39    home  lost+found  opt    resume.initrd  sbin  sys      var
zem@denkbrett:~$
```

Und schon kennen wir das Komando *ls* (list) das uns den Inhalt eines Verzeichnisses anzeigt. Der Parameter ist optional. Wenn er nicht angegeben wird wird der Inhalt des aktuellen Arbeitsverzeichnis angezeigt.

Das Arbeitsverzeichnis ist das verzeichnis in dem sich die Komandozeile grade befindet, auf dem sie also arbeitet. Welches Verzeichnis das ist, sagt uns das Komando *pwd* (print working directory)

```
zem@denkbrett:~$ pwd
/home/zem
zem@denkbrett:~$
```

Jedem Systembenutzer ist ein Heimatverzeichnis (home directory) zugeordnet. Das Heimatverzeichnis kann auch durch die ~ oder mit $HOME abgekürzt werden. Nach dem Login, ist das Arbeitsverzeichnis der Shell auf das Homeverzeichnis des eingeloggten Benutzers eingestellt, jeder Benutzer hat so ein Homeverzeichnis. Der Standard empfielt das das $HOME eines Benutzers ein unterverzeichnis von /home/ ist und den gleichen namen bekommt wie der Loginname also /home/$user.

Im $HOME hat der jeweilige Benutzer schreibrechte um Konfigurationsdaten, eigene Programme und Dateien abzulegen. Konfigurationsdateien und verzeichnisse die Konfigurationsdateien enthalten im Homeverzeichnis beginnen üblicherweise mit einem . im Dateinamen, da *ls* solche dateien nur anzeigt wenn es mit der option -a aufgerufen wird.

```
zem@denkbrett:/home/e17zem$ ls
desktop  Desktop  Downloads  Vortrag_Wellenoptik_Fourier.pdf
zem@denkbrett:/home/e17zem$ ls -a
.              .bashrc  desktop   Downloads  .gnome2          .lesshst   .profile       .taxbird     Vo
..             .cache   Desktop   .e         .gnome2_private  .local     .pulse         .thumbnails  .X
.bash_history  .config  .dfmdesk  .gconf     .gstreamer-0.10  .mozilla   .pulse-cookie  .tkdesk      .x
.bash_logout   .dbus    .dmrc     .gconfd    .gvfs            .nautilus  .Skype         .vfu
zem@denkbrett:/home/e17zem$
```

Standardmäßig zeigt ls nur die einträge an, aber nicht ob es sich um eine Datei oder ein Verzeichnis handelt[7] der parameter für die Detailausgabe mit Zusatzinfos zu den Datzeien *-l* zeigt uns mehr.

```
zem@denkbrett:/home/e17zem$ ls -l -a
insgesamt 412
drwxr-xr-x 25 e17zem e17zem   4096 Okt 19 20:58 .
drwxr-xr-x  4 root   root     4096 Nov 16  2010 ..
-rw-------  1 e17zem e17zem    998 Okt 19 21:14 .bash_history
-rw-r--r--  1 e17zem e17zem    220 Apr 30  2010 .bash_logout
-rw-r--r--  1 e17zem e17zem   3187 Apr 30  2010 .bashrc
drwxr-xr-x  5 e17zem e17zem   4096 Okt 19 20:48 .cache
drwxr-xr-x  4 e17zem e17zem   4096 Okt 19 20:48 .config
drwx------  3 e17zem e17zem   4096 Apr 30  2010 .dbus
drwxr-xr-x  2 e17zem e17zem   4096 Apr 30  2010 desktop
drwxr-xr-x  2 e17zem e17zem   4096 Nov 16  2010 Desktop
drwxr-xr-x  2 e17zem e17zem   4096 Apr 30  2010 .dfmdesk
-rw-------  1 e17zem e17zem     34 Nov 16  2010 .dmrc
drwx------  2 e17zem e17zem   4096 Okt 19 20:48 Downloads
drwxr-xr-x  3 e17zem e17zem   4096 Apr 30  2010 .e
```

[7] Vorraussgesetzt wir ignorieren die farblich blaue Hervorhebung der Verzeichnisse, es passiert aber auch schnell mal das das Terminal farbliche hervorhebungen ignoriert

```
drwx------  3 e17zem e17zem   4096 Okt 19 20:58 .gconf
drwx------  2 e17zem e17zem   4096 Okt 19 20:59 .gconfd
drwx------  6 e17zem e17zem   4096 Okt 19 20:48 .gnome2
drwx------  2 e17zem e17zem   4096 Apr 30  2010 .gnome2_private
drwxr-xr-x  2 e17zem e17zem   4096 Okt 19 20:48 .gstreamer-0.10
drwx------  2 e17zem e17zem   4096 Apr 30  2010 .gvfs
-rw-------  1 e17zem e17zem     53 Apr 30  2010 .lesshst
drwxr-xr-x  3 e17zem e17zem   4096 Apr 30  2010 .local
drwx------  4 e17zem e17zem   4096 Okt 19 20:58 .mozilla
drwxr-xr-x  2 e17zem e17zem   4096 Okt 19 20:48 .nautilus
-rw-r--r--  1 e17zem e17zem    675 Apr 30  2010 .profile
drwx------  2 e17zem e17zem   4096 Okt 19 20:34 .pulse
-rw-------  1 e17zem e17zem    256 Okt 19 20:34 .pulse-cookie
drwx------  5 e17zem e17zem   4096 Okt 19 20:57 .Skype
drwx------  2 e17zem e17zem   4096 Apr 30  2010 .taxbird
drwx------  3 e17zem e17zem   4096 Okt 19 20:48 .thumbnails
drwxr-xr-x  3 e17zem e17zem   4096 Apr 30  2010 .tkdesk
drwxr-xr-x  2 e17zem e17zem   4096 Apr 30  2010 .vfu
-rw-r--r--  1 e17zem e17zem 276906 Okt 19 20:47 Vortrag_Wellenoptik_Fourier.pdf
-rw-------  1 e17zem e17zem     54 Okt 19 20:34 .Xauthority
-rw-r--r--  1 e17zem e17zem   7606 Nov 16  2010 .xsession-errors
zem@denkbrett:/home/e17zem$
```

Das d ganz vorne am Eintrag zeigt an ob es sich um eine Datei oder ein Verzeichnis handelt. $d = directory$. Auffällig sind auch die beiden Sonderverzeichnisse . und .. Dabei handelt es sich bei . um einen verweis auf das aktuelle verzeichnis und bei .. um einen Verweis auf das verzeichnis eine ebene höher, also in unserem Beispiel /home.

Pfade kann man auf zwei arten angeben:

- relativ
- absolut

Ein absoluter pfad beginnt immer mit / also im root verzeichnis. Sämtliche Verzeichnisse bis zum Speicherort der Datei müssen mit angegeben werden. Das ist insbesondere dann nützlich wenn man nicht weis wo man sich befindet oder der weg von / ausgehend kürzer ist.

Ein relativer pfad beginnt nie mit /. in diesem Fall ist die angabe relativ zum arbeitsverzeichnis zu sehen. Also

$$\text{Arbeitsverzeichnis} + \text{Relativer Pfad} = \text{Absoluter Pfad}$$

Übung Nehmen wir an das Arbeitsverzeichnis ist */home/zem* wie lautet der relative Pfad zur Datei */var/log/messages*?

6.1 Dateinamen

Dateinamen können beliebig lang sein, und sind case sensitiv. Leerzeichen sind zwar erlaubt, allerdings können wir bereits absehen das Dateinamen die Leerzeichen enthalten nicht unbedingt einfach zu verarbeiten sind.

6.2 Wichtige Verzeichnisse

/etc systemweite Konfigurationsdateien

/bin anwendungsprogramme die für den Betrieb des Systems notwendig sind

/sbin administrationsprogramme die für den Betrieb des Systems notwendig sind

/lib bibliotheken die für den Betrieb des Systems notwendig sind

/dev virtuelles Dateisystem für Gerätedateien

/proc virtuelles Dateisystem für Prozessinformationen

/sys virtuelles Dateisystem für Systemstatusinformationen

/tmp Verzeichnis für temporäre dateien

/root homeverzeichnis des Systemadministrators (root)

/media mountpoints für wechselmedien

/mnt mountpoints für wechselmedien

/usr Verzeichnis für die Installation von Anwendungen die nicht für den Betrieb des Systems erforderlich sind.

/var Verzeichnis für Datenbanken und Anwendungsdaten, Logfiles etc. . .

7 Arbeitsverzeichnis wechseln

Das Arbeitsverzeichnis können wir mit dem komando **cd** *zielverzeichnis* wechseln. cd ohne weiteren Parameter wechselt ins Heimatverzeichnis.

8 manpages

„If you teach a man to fish he can eat, if you teach a fish to man he can unix“

Kommen wir jetzt zum wichtigsten Kommando, nämlich dem, das die Hilfe für ein Komando öffnet.

man *Hilfeseite*

Öffnet die passende Hilfeseite. Mit q können wir die Ansicht wieder verlassen.

Wenn das exakte Komando nicht bekannt ist oder nicht gefunden werden kann, können wir mit auch dem Komando apropos nach Manpages suchen die das Thema behandeln.

Übung Suche nach dem Komando um ein neues Verzeichnis anzulegen

Übung Was macht ls -g -o -t -a

9 Weitere wichtige Kommandos

rmdir Löscht ein verzeichnis sofern dieses leer ist

rm Löscht eine datei (meist ohne rückfrage)

rm -r Löscht ein verzeichnis mit inhalt rekursiv

rm -f Löscht immer ohne Rückfrage

mkdir erstellt ein Verzeichnis

touch Ändert den Zeitstempel einer datei, oder legt eine neue Datei an

cat Gibt eine Datei auf der Komandozeile aus

10 Komandoverkettung, Pipes und Ausgabeumleitungen

Wir führen jetzt das Komando *ls -l /etc* aus. Wie wir sehen ist das Komando viel zu lang für den Bildschirm. Wir können jedoch die ausgabe des Komandos mit Hilfe des — symbol (pipe) an die eingabe eines weiteren Programmes leiten, das in der lage ist die Ausgabe seitenweise anzuzeigen. Günstige kandidaten für sowas sind more und less.

```
$ ls -l -a /etc | less
```

Des weiteren besteht noch die möglichkeit die Ausgabe in eine Datei zu schreiben > *Dateiname* oder Daten zur verarbeitung aus einer Datei zu lesen < *Dateiname*.

```
$ echo inhalt einer Testdatei > Testdatei
$ cat Testdatei
inhalt einer Testdatei
$ cat < Testdatei
inhalt einer Testdatei
```

Wenn das Kommando cat ohne Dateinamen aufgerufen wird liest es von der Standardeingabe, das ist entweder die Tastatur oder aber ein anderes Programm. cat kann auch verwendet werden um einen header und einen Footer an eine Datei anzuhängen.

Die beiden Pfeile gibt es auch als Doppelpfeil. Beim Doppelpfeil >> wird an die zu schreibende Datei angehangen, beim einfachpfeil eine neue Datei erzeugt.

```
$ echo neuer inhalt einer Testdatei > Testdatei
$ cat Testdatei
neuer inhalt einer Testdatei
$ echo zweite Zeile der Testdatei >> Testdatei
$ cat < Testdatei
inhalt einer Testdatei
zweite Zeile der Testdatei
```

Zu guter letzt muss noch der Standardfehlerkanal erklärt werden. 2 > und 2 >> Das ist ein extrakanal für Status und fehlermeldungen, damit die bei einer Komandoverkettung nicht zwischen den Daten landen. Hin und wieder kann es auch Hilfreich sein die Produktion von Fehlern zu unterdrücken, das können wir machen indem wir den Fehlerkanal auf die Datei /dev/null umleiten. /dev/null die alle Daten die in sie geschrieben wird vernichtet.

11 find/locate

find findet dateien (dateisuche). Locate macht auch sowas, benutzt dafür aber eine Datenbank, die ist schneller aber nicht unbedingt aktuell.

übung Irgendwo auf dem System ist eine Datei fnord in einem verzeichniss foo. Findet die Datei und kopiert sie in euer homeverzeichnis.

12 grep

Grep holt nur bestimmte Zeilen aus einer Datei und zeigt diese an.

übung Sucht in der Datei fnord eine Zeile die mit TAG beginnt und gebt sie aus.

13 chmod/chown/group/passwd/adduser

- Es gibt unter Linux eine Anzahl von benutzern die durch eine Benutzernummer unterschieden werden. Damit der Anwender sich nicht mit Namen rumärgern muss, bekommt jeder Benutzer auch einen Namen.
- Der Systemadministrator hat die UserID 0 und den Namen root, der darf alles. Es gibt T-Shirts mit der aufschrift „Ich bin /root, ich darf das“
- Einem Benutzer sind immer eine Hauptgruppe und beliebig viele Untergruppen zugeordnet.
- Linux kennt rechte für den Besitzer, die Gruppe und alle anderen.
- mit 3 zusätzlichen spezialrechten zusammen kommt man so auf 14 bit um die dateirechte darzustellen.
- Die Rechte sind rwx (lesen schreiben ausführen) jeweils für den Benutzer, die Gruppe und alle anderen.

- Die Sonderrechte sind:
  Set UserID on Execution
  Set Group ID on Execution
  Sticky bit

14 VI

- Arbeitsmodis
- i
- ESC
- command mode
- suchen /
- :1,$s/foo/bar/g suchen und ersetzen
- d delete
- p put in
- :syntax on

15 Shell Scripte

- sind eine zusammenstellung von konsolen Komandos
- enthalten oft weitere Kontrollstrukturen
- dienen zur Automatisierung

Übung Schellscript schreiben das 5 mal (zeilennummer) Hello World ausgibt, zeilennummer soll dafür nummeriert werden.

16 Umgebungsvariablen

- PATH
- PS1
- HOME

```
$ set
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:expand_aliases:extglob:extquote:force_fignore:interactive_comments:progco
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_COMPLETION=/etc/bash_completion
BASH_COMPLETION_DIR=/etc/bash_completion.d
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="4" [1]="2" [2]="20" [3]="1" [4]="release" [5]="i486-pc-linux-gnu")
BASH_VERSION='4.2.20(1)-release'
COLORFGBG='15;default;0'
COLORTERM=rxvt-xpm
COLUMNS=167
COMP_WORDBREAKS=$' \t\n"\'><=;|&(:'
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-HqyvObySmk,guid=f37dea452bd4ebcee26442900000005d
```

```
DESKTOP_SESSION=default
DIRSTACK=()
DISPLAY=:0
EDITOR='gvim -f'
EUID=1000
GDMSESSION=default
GNOME_KEYRING_CONTROL=/tmp/keyring-CGNmz4
GNOME_KEYRING_PID=3619
GROUPS=()
HISTCONTROL=ignoredups
HISTFILE=/home/zem/.bash_history
HISTFILESIZE=500
HISTSIZE=500
HOME=/home/zem
HOSTNAME=denkbrett
HOSTTYPE=i486
IFS=$' \t\n'
LANG=de_DE.UTF-8
LD_PRELOAD=/usr/lib/scrotwm/libswmhack.so.0.0
LESSCLOSE='/usr/bin/lesspipe %s %s'
LESSOPEN='| /usr/bin/lesspipe %s'
LINES=44
LOGNAME=zem
...
```

- Shell Aliases
- Shell Funktionen
- /etc/profile
- .bash_profile

17 Job Kontrolle

- CTRL+Z
- bg fg jobs
- CTRL+C
- CTRL+D

18 screen

Ist eine Anwendung die Mehrere Terminals auf einer Konsole darstellt.

19 X11

19.1 Window Manager

20 Softwarepackete