Betriebssysteme

Linux Prozesse

Gliederung

- Installation von Ubuntu
- Wichtige Linux Befehle
- Beispiele von Prozessen und ihre Behandlung unter Linux

Installation von Ubuntu in VMware

1. Download the **Ubuntu** iso (desktop not server) and the free **VMware** Player.
2. **Install VMware** Player and run it, you'll see something like this:
3. Select “Create a New Virtual Machine”
4. Select “Installer disc image file” and browse **to** the **Ubuntu** iso you downloaded.

Linux Ubuntu

- Loggen Sie sich in ihr Linux-System ein.
- Machen Sie sich mit dem System und der Umgebung vertraut.
- Befehle, die Sie öfter brauchen:
  - # man < Befehl> // zeigt Ihnen die Manualseite zu dem Befehl an. Beispiel: man ls
  - #ls // Das Komando zeigt die Liste der Files und Verzeichnisse an.
  - #cd // ändern des Verzeichnis
  - #pwd // zeigt den Pfad des momentan Verzeichnisses an.
  - #ps // zeigt die momentane Prozesse
  - #cat <file>// zeigt den Inhalt des <file> an
  - #nano <file> //nano ist ein kleiner Editor
  - #mkdir <Verzeichnis> // ein Verzeichnis erstellen
  - #rmdir < Verzeichnis> // Verzeichnis löschen
  - #mv <file> <Verzeichnis> // ein <file> in ein Verzeichnis bewegen
  - #cp <file> <Verzeichnis> // ein <file> in ein Verzeichnis kopieren
  - #rm <file> // ein <file> löschen

Installation von Programmen

- Das Paketverwaltungssystem unter Ubuntu heißt:
- #apt
- #apt –v // zeit die Version von apt an
- Installation des C bzw. C++ Compilers
- #sudo apt install g++ bzw.
- #sudo apt install build-essential

Administration

- Die Gruppen anzeigen, die zu einem user gehören:
  - #groups <user>
- Neue Gruppe und User anlegen
  - Schauen Sie sich die Beschreibung von addgroup und adduser an.
  - #man adduser
  - #man addgroup
  - Was steht in /etc/adduser.conf
- Unter einen anderen user anmelden
- #su <user> // (set user)
- #exit // beenden der Shell
- #shutdown –r now // Reboot des Systems

Administration

User und Gruppen anlegen

- Neue Gruppe und User anlegen
  - #sudo addgroup dhbw
  - #sudo adduser –-ingroup dhbw fschmatz
- Passwort und Shell ändern, falls gewünscht
  - #passwd fschmatz (Passwort festlegen)
  - #chsh –s /bin/sh
- Gruppe und User entfernen
  - #groupdel dhbw
  - #userdel fschmatz bzw.
  - #userdel –r fschmatz (user und home-Verzeichnis entfernen)
- Schauen Sie sich das File /etc/passwd an.
- Was steht da drinnen?

Administration

Speicher und Filegrößen

- Speichercheck
  - #df –h // Diskfile Speichernutzung
  - #man df
- Größe eines Verzeichnis
  - #du –sk /home
  - Option
    - s: Aufsummierte Totaler Speicherplatz
    - k: Angabe in Kilobytes
  - #man du

Prozesse in Linux anzeigen

- Schauen Sie sich die Manualseiten des Kommandos ps an.
  - # man ps
- Wichtige Options:
- #ps
- #ps –ax
- #ps –alx
- #pstree

Prozesserzeugung unter Unix/Linux

Prozesserzeugung - fork

- Ein Prozess wird unter Unix durch einen ***fork()***-Aufruf des Vaters erzeugt
- Der Kindprozess erzeugt und erbt dessen Umgebung **als Kopie**:
  - Alle offenen Dateien und Netzwerkverbindungen
  - Umgebungsvariablen
  - Aktuelles Arbeitsverzeichnis
  - Datenbereiche
  - Codebereiche
- Durch den System-Call ***execve()*** kann im Kindprozess ein neues Programm geladen werden

Beispiel einfaches fork

```
#include <unistd.h>
#include <stdio.h>
int main() {
        pid_t id = fork();
        printf("Process ID: %d\n", id);
        return 0;
}
```

Wo befinden Sie die include-Files?
Schauen Sie unter /usr/include und geben Sie das Headerfile aus:
cat /usr/include/uinstd.h | more

Beispiel fork mit Abfrage der id

```
#include <unistd.h>
#include <stdio.h>
int main(int argc, char** args)
{
pid_t id;
switch(id = fork())
                {
                case -1: /*Fehler beim fork */
                printf("Fehler ... \n");
                break;
                case 0: /* Kindprozess */
                printf("Ich bin das Kind mit ID:%d\n",id);
                break;
                default: /*Erzeuger */
                printf("Ich bin der Ezeuger ID:%d\n",id);
                break;
                }
return 0;
}
```

Was wird dem Kind vererbt

- Arbeits- und Wurzelverzeichnis
- Dateierstellungsmaske
- Signalmaske und Handler
- UID, EUID, GID, EGID, Prozessgruppen-, Session-, Setuser- und Setgroup-ID etc.
- Steuerterminal (CTTY)
- Umgebungsvariablen
- Ressourcenlimits
- Shared-Memory-Segmente (sofern verwendet)

Was bekommt das Kind nicht?

- Prozess-ID (PID) und Elternprozess-ID (PPID)
- die Dateisperren (Lockings)
- Zeitschaltuhren, die z. B. mit alarm() gesetzt wurden
- noch nicht ausgeführte Signale (hängende Signale)
- Die Zeiten utime, stime, cutime und ustime werden beim Kindprozess auf 0 gesetzt.

Aufgabe multiple fork()

- Wieviel Prozesse werden erzeugt?

```
#include <stdio.h>
#include <sys/types.h>
int main()
{
        fork();
        fork();
        fork();
        printf("hello\n");
        return 0;
}
```

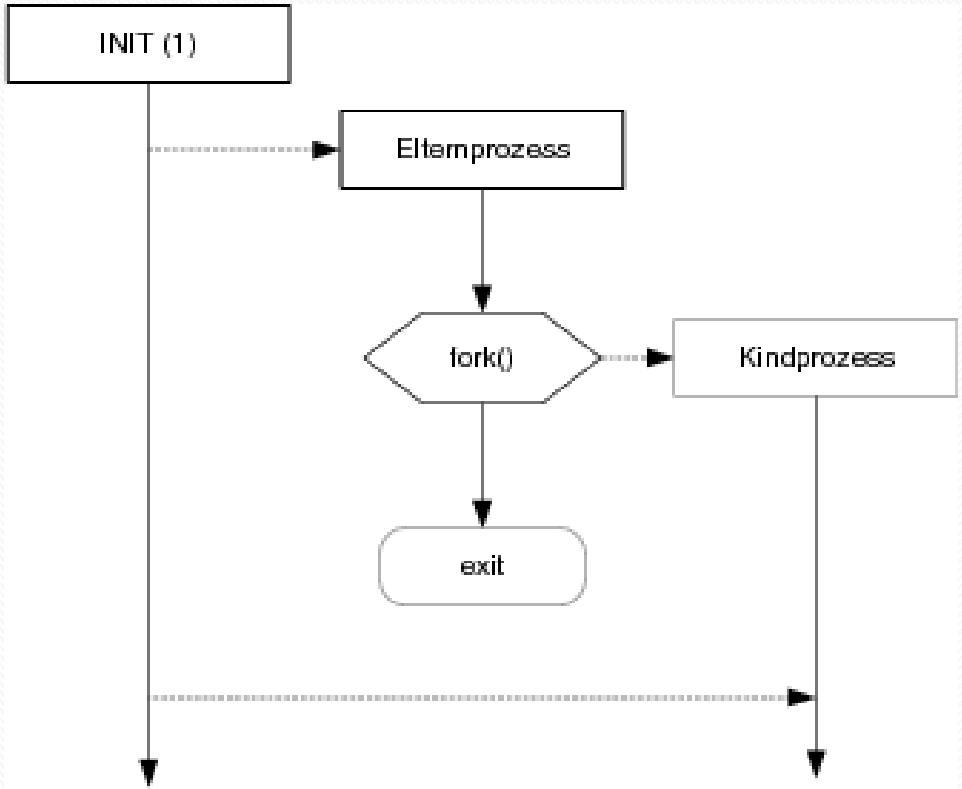
Probleme bei der Prozesserzeugung

- Das Kind hat plötzlich den Elternteil verloren (Verwaisen)
- Das Elternteil hat das Kind verloren (Zombie)

Kind verwaist

```
/* waise.c */
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
int main (void) {
  pid_t pid;
  switch (pid = fork ()) {
  case -1:
    printf ("Fehler bei fork()\n");
    break;
  case 0:
    printf ("--- Im Kindprozess (%d) ---\n",getpid());
    printf("Elternprozess : %d\n", getppid());
    sleep(2);
    printf ("--- Im Kindprozess (%d) ---\n",getpid());
    printf("Elternprozess : %d\n", getppid());
    break;
  default:
    printf ("--- Im Elternprozess (%d) ---\n",getpid());
    sleep(1);
    printf(" --- Elternprozess endet vorzeitig ---\n");
    exit(0);  /* Sofort beenden */
  }
  return EXIT_SUCCESS;
}
```

Warten Kind verwaist.



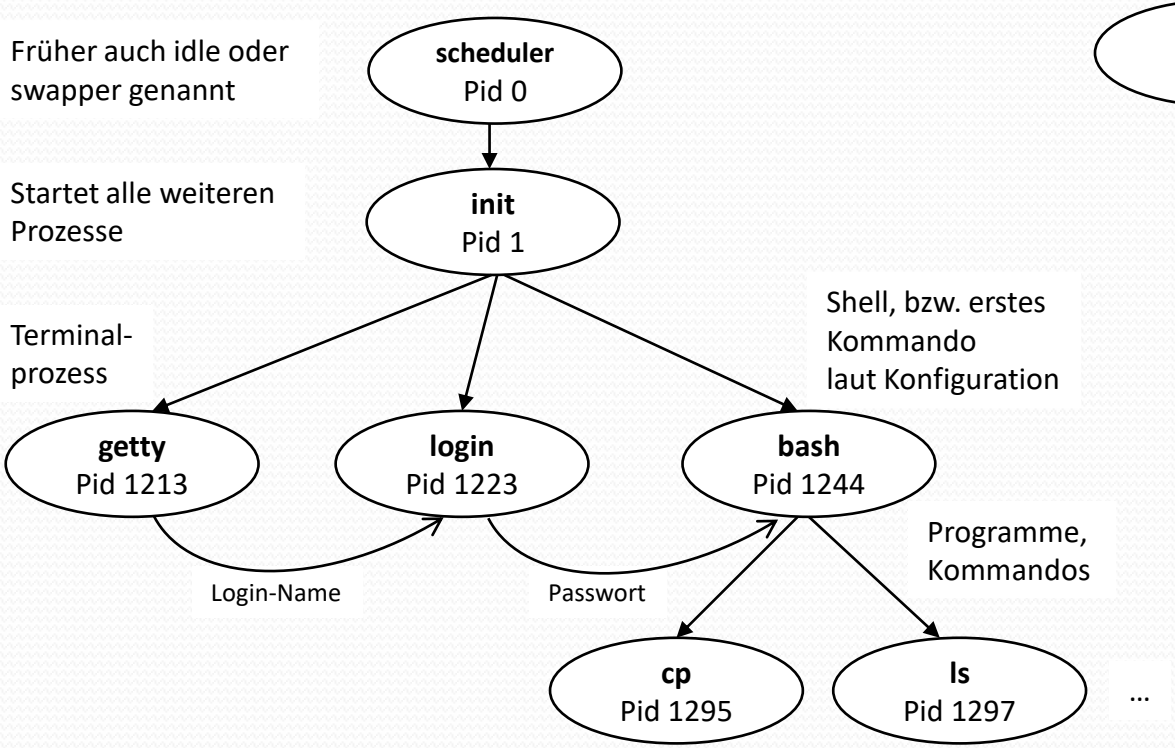
Zombie

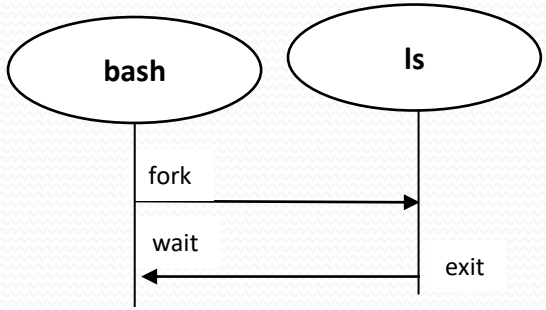
Kind endet und das Elternteil weiß das nicht

```
/* zombie.c */
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
int main (void) {
  pid_t pid;
  switch (pid = fork ()) {
  case -1:
    printf ("Fehler bei fork()\n");
    break;
  case 0:
    printf("--- Im Kindprozess (%d) ---\n",getpid());
    printf("Elternprozess : %d\n", getppid());
    printf("--- Kindprozess beendet sich ---\n");
    exit(0);
  default:
    printf("--- Im Elternprozess (%d) ---\n",getpid());
    printf("--- Beenden mit <STRG>+<C> ---\n");
    while(1); /* Endlosschleife */
  }
  return EXIT_SUCCESS;
}
```

Unix-Prozessbaum

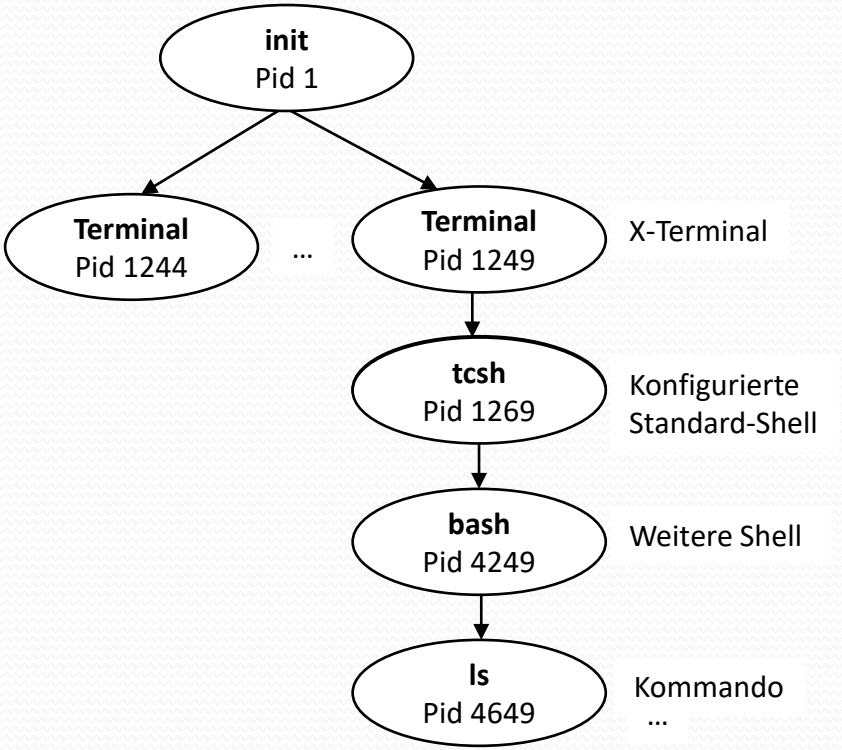
- Je Terminal wartet ein getty-Prozess auf eine Eingabe (Login)
- Nach erfolgreichem Login wird ein Shell-Prozess eröffnet
- Jedes Kommando wird gewöhnlich in einem eigenen Prozess ausgeführt





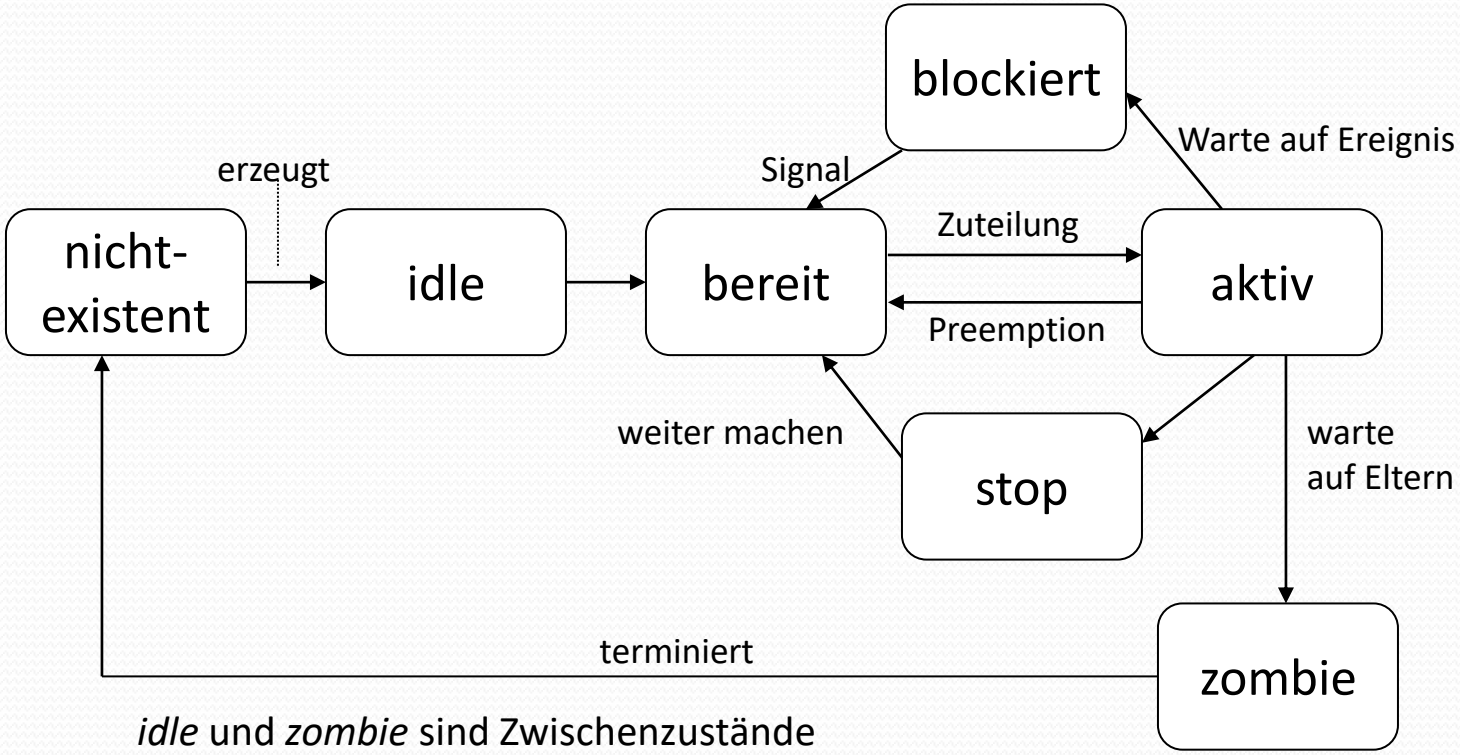
Unix-Prozessbaum

- Prozesssicht nach dem Login: Kommando pstree
- Ein Prozess mit Bezeichnung **Terminal** als X-Terminal (Terminal-Emulation unter grafischer Oberfläche) läuft



Zustandsautomat eines Unix-Prozesses

- Jeder Prozess, außer der init-Prozess, hat einen Elternprozess
- Zustand *zombie* wird vom Kindprozess eingenommen, bis der Elternprozess Nachricht über Ableben erhalten hat
- Elternprozess stirbt vorher –> init-Prozess wird „Pflegevater“



idle und *zombie* sind Zwischenzustände