Betriebssysteme

Übungen 06

Prof. Dr. Rainer Werthebach

Studiengang Informatik

Hochschule Aalen - Technik und Wirtschaft

Prozesse

- `ps` → Prozessinformationen anzeigen

  - wichtige Parameter:
    `-aux` → alle Prozesse (auch Daemons) mit ausführlichen Informationen
    `-A` oder `-e` → alle Prozesse (auch Daemons) mit weniger Informationen

  - Beispiel:
    ```
    user@recher ~ $ ps -aux | less
    ```

Hintergrundprozesse

- `Kommando &` → Prozess im Hintergrund starten
- `fg %ProzessNR` → bringt Prozess in den Vordergrund
  - Beispiel: `user@rechner ~ $ fg %2`
- `bg %ProzessNR` → zurück in den Hintergrund
  - um Prozess anzuhalten und Prompt zu bekommen, `<strg>+<z>` drücken
  - wenn keine `ProzessNR` angegeben wird, wird aktueller Prozess genommen
- `jobs` → Liste der mit `Kommando &` ausgeführten Programme

Signale

- Signale können Prozesse ohne Eingabemöglichkeit steuern

- typisches Auftreten:
  - durch Eingabe des Benutzers (User-Interrupts)
  - illegales Kommando bzw. Argument
  - fehlerhafte Ausführung des Programms
  - Hardwarefehler
  - Ende eines Kindprozesses

- Tabelle mit allen Signalbezeichnungen und Nummern unter:
  - http://de.wikipedia.org/wiki/Signal_(Computer)

Signale senden

- `kill [-SignalNR] ProzessNR`

  → Beendet einen Prozess (SIGTERM) „oder“
  sendet ihm ein beliebiges Signal

  - Wenn -SignalNR nicht angegeben wird, wird SIGTERM (-15) gesendet
  - Beispiele:
    - `user@rechner ~ $ kill 5320`
    - `user@rechner ~ $ kill -9 4522`
    - `user@rechner ~ $ kill -2 6312`
    - `user@rechner ~ $ kill -10 4023`

Signale abfangen

- Syntax:
  - `trap 'Komandoliste' Signal1 [Signal2 …]`
- bei nicht unterbrechenden Signalen wird aktuelles Kommando erst fertig ausgeführt und dann die Kommandoliste von `trap` abgearbeitet
- Beispiele:
  - `trap 'echo Programm mit <Strg>+<c> (SIGINT) beendet; exit' 2`
  - `trap 'echo SIGUSR1 oder SIGUSR2 abgefangen' SIGUSR1 12`
  - `trap '' 2`