Zusammenfassung Tag 14

Prozesse und Ressourcenverbrauch anzeigen

- Der Befehl ps ist vielfältig und kann in unterschiedlicher Notation genutzt werden:
  - Linux-Notation: Optionen werden mit Minus (-) eingeleitet
  - BSD-Notation: Optionen werden ohne Minus geschrieben
  - GNU-Notation: Doppeltes Minus als Einleitung, danach komplette Wörter
- Insbesondere Linux- und BSD-Notation nutzen unterschiedliche Buchstaben für dieselben Optionen, z.B. ps -e = ps ax
- Welche Bedeutung die einzelnen Spalten haben, zeigt man ps
- Der Befehl top zeigt die Prozesse und deren Ressourcennutzung dynamisch an
- Mit htop wird die Darstellung noch etwas schöner
- Beide Tools ermöglichen die Darstellung nach Spalten, htop unterstützt zudem Filter
- Mit pstree können die Prozesse hierarchisch angezeigt werden

Programme im Vordergrund und im Hingergrund ausführen und verwalten

- Programme sperren normalerweise das Terminal, von dem aus sie aufgerufen wurden
- Programme können mit STRG+C beendet werden
- Programme können mit & im Hintergrund gestartet werden, sodass das Terminal weiter nutzbar bleibt
- Jobs können mit fg und bg in den Vordergrund und zurück verschoben werden
- Wird die Shell beendet, werden auch alle in dieser Shell gestarteten Programme beendet
- Mit nohup <Prozess> kann ein Prozess so gestartet werden, dass er das HUP-Signal ignoriert, wenn die Shell beendet wird. Seine Ausgaben werden in nohup.out geschrieben
- Mit disown <BG-ID> kann ein Prozess nachträglich von der Shell getrennt werden

Prozesse beenden

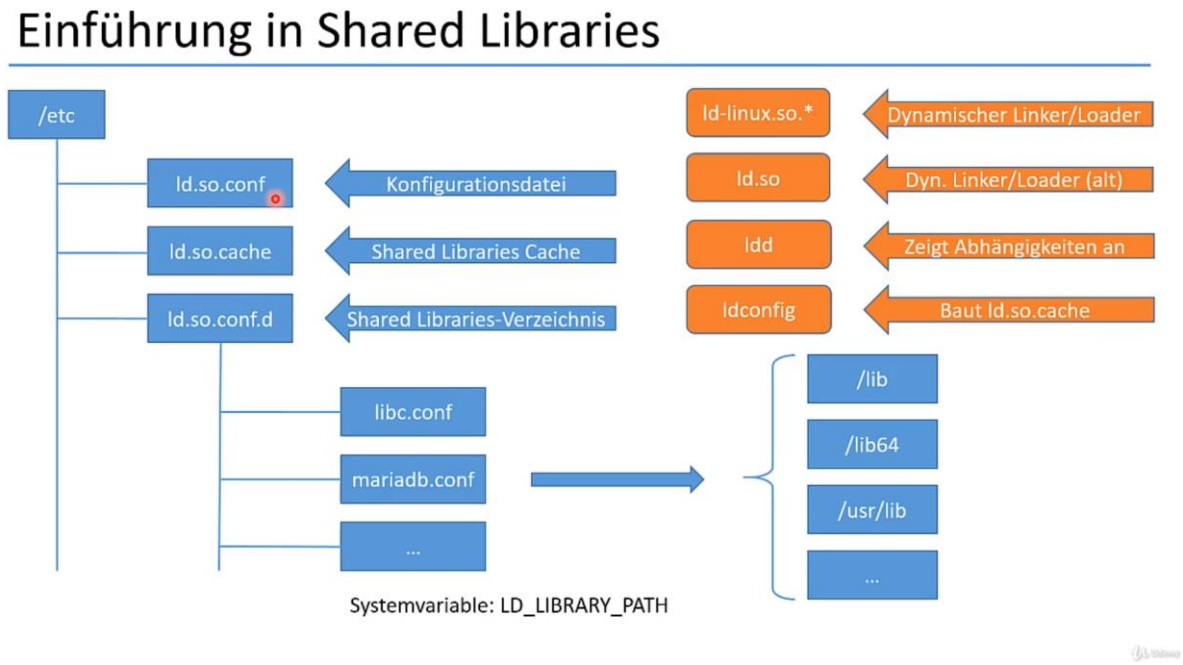
- Jeder Prozess hat eine automatisch vom Betriebssystem zugewiesene PID
- Mit dem Befehl kill können diverse Signale an die Prozesse gesendet werden:
  - SIGTERM (15): normale Beendigung eines Prozesses
  - SIGKILL (9): harte Beendigung des Prozesses
  - SIGINT (2): Wird von STRG+C gesendet, wirkt wie SIGTERM
  - SIGSTOP (19): Wird von STRG+Z gesendet, hält den Prozess an
  - SIGHUP (1): Sendet das Terminal beim Beenden
- Mit killall <Zeichenkette> können alle Prozesse beendet werden, die der Zeichenkette entsprechen

Prozess-Prioritäten mit Nice-Levels steuern

- Jedem Prozess wird vom Linux-System bzw. der CPU ein bestimmter Grad an Aufmerksamkeit zugestanden
- Dieser Grad wird durch das Nice-Level bestimmt, Standard ist 0
- Die Priorität liegt zwischen -20 (höchste) und +19 (niedrigste)
- Prioritäten können mit dem Programm nice beim Start des Programms gesetzt werden
- Nur root darf Nice-Level heruntersetzen, Programme also höher priorisieren
- Ein Programm mit Nice-Level -20 kann das System beeinträchtigen, dies sollte nur sehr vorsichtig eingesetzt werden
- Mit renice können die Prioritäten eines laufenden Prozesses angepasst werden
- Nice-Levels sind keine Booster-Funktion und können nur als eines von mehreren Tuning-Mitteln gesehen werden

Einführung in die Shared Libraries

- Shared Libraries (Programmbibliotheken) sind Routine-Funktionen, die immer wieder benötigt werden und von verschiedenen Programmen auf dem System genutzt werden können
- Vorteil: Es müssen nicht immer alle Funktionen in jedes Programm einkompiliert werden
- Nachteil: Die Shared Libraries müssen zur Verfügung stehen, sonst läuft das Programm nicht (Abhängigkeiten)
- Es existieren diverse Pfade und Programme für die Verwaltung von Shared Libraries



Shared Libraries in der Praxis

- /etc/ld.so.conf ist die Hauptkonfigurationsdatei für die Verwaltung der Programmbibliotheken
- Unter /etc/ld.so.conf.d können Dateien mit Pfaden zu Shared Libraries hinterlegt werden

- /etc/ld.so.cache enthält die Pfade aller bekannten Shared Libraries in binärer Form
- Pfade zu Bibliotheken können auch über die Systemvariable $LD_LIBRARY_PATH gesetzt werden, ihre Syntax ist analog zur Variable PATH
- Mit ldd <Programmpfad> können die Shared Libraries, von denen ein Programm abhängig ist, angezeigt werden
- Mit ldconfig kann der Cache erstellt werden und neue Libraries registriert werden