UNIX-Werkzeuge

3. Hilfsmittel für die Compilierung und den Test

Allgemein

```
gcc - GNU project C und C++ Compiler

   Kompiliert ein oder mehrere C-Quelltexte oder Assembler-Quellen.
   gcc ruft automatisch alle Schritte auf, die notwendig sind um ein
   ausführbares Programm zu erzeugen, wenn dies nicht explizite unter-
   bunden wird. gcc benutzt dabei folgende Standard-Suffixe:
          .c  - C-Quelltext
          .i  - C-Quelltext nach Aufbereitung durch den Preprozessor
          .s  - Assemblerquelltext
          .o  - Objektmodul
   Es existieren folgende Phasen der Compilierung:
         Preprozessor  -  gcc -E   (nur Quellkode aufbereitung)
         Compiler      -  gcc -S   (nur Assemblercode erzeugen)
         Assembler     -  gcc -c   (nur Ojektmodul erzeugen)
         Linker        -  gcc      ladbares Programm erzeugen
   gcc kennt sehr viele Optionen!!!!
   Optionen nicht gruppierbar,
          also -d -r ist nicht identisch mit -dr
   Reihenfolge von Bedeutung, wenn mehrmals die gleiche Option mit
   unterschiedlicher Belegung vorkommt
             -L/usr/lib -L/usr/local/lib
```

```
Manual (4.8.1)enthält 14945 Zeilen!!!!! Über 200 Seiten!!!!
gcc --help   gibt einen ersten Anhaltspunkt
   Usage: gcc [options] file...
   Options:
     -pass-exit-codes         Rückgabe des höchste Fehlercodes
     --help                   Help-Informationen (60 Zeilen)
     --target-help            Help für Zieloptionen
     -v --help                Alle Helpinformationen (3875 Zeilen)
     -dumpspecs               Alle Betriebssystemabhängigkeiten
     -dumpversion             Compilerversion
     -dumpmachine             Zielsystem
     -print-search-dirs       Suchpfade
     -print-libgcc-file-name  allgemeine Compiler-Bibliothek
     -print-file-name=<lib>   Anzeigen Pfad zu <lib>
     -print-prog-name=<prog>  Anzeigen Pfad zu <prog>
     -Xassembler <arg>        Pass <arg> zum Assembler
     -Xpreprocessor <arg>     Pass <arg> zum Preprocessor
     -Xlinker <arg>           Pass <arg> zum Linker
     -combine                 Compiliere mehrere Quellen als eine
     -save-temps              Rette TMP-Files
     -pipe                    Benutze Pipes
     -time                    Zeitangaben
     --sysroot=<directory>    root Directory für Headers und Bibliotheken
     -B <directory>           Add <directory> to the compiler's search pat
     -b <machine>             Zielcode <machine>, wenn installiert
     -V <version>             Run gcc version number <version>
     -v                       Verbose
     -E                       nur Preprocessor
     -S                       nur Compiler; kein Assembler oder Linker
```



```
     -c                       Compiler und Assembler, nicht linken
     -o <file>                Ausgabe-File
     -x <language>            Eingabesprache:
                               c, c++, assembler, none
     -ansi                    ANSI-Standard

   Optionen, die mit -g, -f, -m, -O, -W, oder --param beginnen werden
   automatisch an das entsprechende Programm weitergeleitet.

Weitere wichtige Optionen:

   -g <nummer>          einfügen von Debugger-Informationen
                        <nummer> : 1, 2, 3
   -include <datei>     Datei einfügen
   -l<lib>              Linken mit Library <lib>
                           -lsocket  libsocket.a, libsocket.so
                           -lnsl     libnsl.a, libnsl.so
   -L<lib-path>         zusätzliches Bibliotheksverzeichnis
   -m32, --32           32-Bit-Architektur
   -m64, --64           64-Bit-Architektur
   -static              statisch linken
   -dynamic             dynamisch linken (standard)
   -O<nummer>           Optimierung
                        <nummer> : 1, 2, 3, 4, ..
                        Vorsicht!! Bis 2 funktioniert es immer
```

```
cpp - Preprozessor für C-Code (Teil von gcc)

   Liefert die gleiche Ausgabe wie gcc -E
   Optionen:  cpp --help (identisch mit gcc --help)

       gcc -E -o hello.i  hello.c
       cpp  -o hello.i  hello.c

as - Assembler, Objektdateien erzeugen

   existiert als selbstständiges Programm oder Teil des gcc.
   (SUN bis Solaris 10: /usr/ccs/bin/as)
   Erzeugt gleiche Ausgabe wie gcc -c
   Optionen:  as --help   (Teil von gcc --help)

        gcc -S -o hello.s  hello.c
        gcc -c -o hello.o  hello.s
                  oder
        as -o hello.o hello.s

ld - Linker - erzeugen von ausführbaren Programmen

   Existiert als selbstständiges Programm oder Teil des gcc.
   (SUN bis Solaris 10: /usr/ccs/bin/ld)
   Optionen: ld --help   (gcc --target-help)
   Erzeugt gleiche Ausgabe wie gcc

      gcc -o hello hello.o
      ld  -o hello hello.o -lc   (denkste)
```



```
size - Auflisten der Größe von Code- und Datensegment
       eines Programmes

Anzeigen der Größe der einzelnen Sektionen eines Binary-Files.

   size [<Option>] [File] ...

     -A              sysv-Format (lang)
     -B              Berkley-format (kurz, standard)
     -o              Zahlenformat octal
     -d              Zahlenformat dezimal
     -x              Zahlenformat hexadezimal
     -t              Summenangaben bei mehreren Files
     -h, --help      Hilfe ausgeben
     -v, --version   Ausgabe Programmversion

      size -B hellod
         text    data     bss     dec     hex filename
          941     272       4    1217     4c1 hellod
```

ldd - Ausgabe der Liste der benutzten Shared Libraries eines Programms

ldd gibt eine Liste aller benötigten Shared Libraries und der zugeordneten Libraries für die angegeben Programme aus. Nicht gefundenen Libraries werden angezeigt.

```
ldd [Optionen] <File> ...

      --help              Hilfstext ausgeben
      --version           Version ausgeben
  -d, --data-relocs       process data relocations
  -r, --function-relocs   process data and function relocations
  -u, --unused            nur Ausgabe der nicht gefunden Libraries
  -v, --verbose           Lange Ausgabe

   ldd hellod
        linux-gate.so.1 =>  (0xffffe000)
        libc.so.6 => /lib/libc.so.6 (0xb7e20000)
        /lib/ld-linux.so.2 (0xb7f75000)
```


nm - Auflisten der Symbole von Objektdateien

Für jedes Symbol in einem Objektfile zeigt nm folgende Werte an:

- **den Wert(Adresse) des Symbols (hexadezimal oder dezimal)**
- **den Typ des Symbols (Kleine Buchstaben - lokale Definition, große Buchstaben - externe Definition):**

```
A  - absoluter Wert (nicht veränderbar)
B  - Wert nicht initialisiert - im Stacksegment
C  - gemeinsames Symbol (mehrfach, wird gelinkt)
D  - Vorinitialisierte Variabel im Datensegment
G  - kleine Variable im Datensegment initialisiert
I  - indirekte Referenz
N  - Debugging-Symbol
R  - Readonly Variable im Datensegment
S  - kleine Variable im Datensegment uninitialisiert
T  - Variable im Textsegment
U  - Variable ist undefiniert (keine Adresse zugeordnet)
W  - "Weaked" Symbol (privat oder durch System definiert)
?  - unbekannter Type
```

- **Name des Symbols**

```
nm [<Optionen>] [<File>] ...

 a.out ist standard für <File>

Optionen:
  -a, --debug-syms        nur Anzeigen von debug-Symbolen
  -A, --print-file-name   Filenamen vor jedes Symbol anzeigen
  -B                      bsd-Format
  -D, --dynamic           nur Anzeigen von dynamischen Symbolen
      --defined-only      nur Anzeigen von definierten Sybolen
  -f, --format=FORMAT     Formatierte Ausgabe: bsd, sysv oder posix
                          Standard:  bsd
  -g, --extern-only       nur Anzeigen von  externen Symbolen
  -l, --line-numbers      Zeilennummern mit angeben, wenn vorhanden
  -n, --numeric-sort      Ausgabe nach Adressen sortieren
  -p, --no-sort           nicht sortieren
  -P, --portability       Ausgabe im Posix-Format
  -r, --reverse-sort      Ausgabe reverse sortieren
  -S, --print-size        Ausgabe der Größe von definierten Sysmbolen
  --size-sort             Ausgabe nach Sort symbols by size
      --special-syms      Include special symbols in the output
      --synthetic         Display synthetic symbols as well
  -t, --radix=RADIX       Zahlenformat, RADIX: oct, dec
  -u, --undefined-only    nur Ausgabe von undefinierten Symbolen
  -h, --help              Hilfe
  -V, --version           Version
```



```
strip - Entfernen von Symbolen aus Programmen

strip entfernt alle Symbole eines Objektfiles und speichert das neue
Objektfile unter dem gleichen Namen wie das alte.

strip [<Optionen>] <File>
  Optionen:
  -I --input-target=<bfdname>      Format des Eingabefiles
  -O --output-target=<bfdname>     Format des Ausgabefile
  -F --target=<bfdname>            Format
  -p --preserve-dates              übernehme Datum und Uhrzeit
  -R --remove-section=<name>       Lösche Section <name> von der Ausgabe
  -s --strip-all                   alles (standard)
  -g -S -d --strip-debug           Lösche alle Debugging Symbole
     --strip-unneeded              Lösche alle Symbole, die nicht für die
                                   Verschiebung gebraucht werden
     --only-keep-debug             Lösche alles außer Debug-Informationen
  -N --strip-symbol=<name>         Entferne das Symbol <name>
  -K --keep-symbol=<name>          Entferne das Symbol <name> nicht
     --keep-file-symbols           Entferne keine File-Symbole
  -w --wildcard                    Erlaube Wildcards in Symbolnamen
  -x --discard-all                 Entferne alle non-global Symbole
  -X --discard-locals              Entferne alle vom Compiler erzeugten Symbo
  -v --verbose                     Verbose
  -V --version                     Anzeigen der Programmversion
  -h --help                        Hilfe
     --info                        Hilfe, Ausgabe aller Architekturinformatio
  -o <file>                        Ausgabe in File <file>
```

ar - verwaltet Funktionsbibliotheken

Mit ar werden beliebig viele Dateien zu einer einzigen zusammengefasst. Eine Kompression findet nicht statt.
ar wird vor allem zur Verwaltung von Bibliotheken für den C-Compiler verwendet. Der Linker benötigt die spezielle Datei __.SYMDEF, in der die Symboltabellen aller Objektdateien zusammengefasst sind.

```
ar [-]{dmpqrstx}[abcfilNoPsSuvV] [Mitglieds-Name] [Zähler] Archiv File...
```

Archiv ist der Name der zu bearbeitenden Archivdatei. Die Archivdateien haben per Konvention die Endung .a. File ist der Name einer Datei, die in das Archiv eingefügt oder aus ihm gelöscht oder extrahiert werden soll.

Kommandos:

```
  d            - Streichen eines Files aus dem Archiv
  m[ab]        - Verschieben eines Files im Archiv (a - after, b -before)
  p            - Ausgabe aller Files
  q[i]         - anfügen an das Archiv
  r[ab][f][u]  - ersetzen
  s            - neues Archiv erzeugen
  t            - Inthaltsverzeichnis
  x[o]         - extrahieren eines  Files aus dem Archiv
```


Kommando-Spezifikation:

```
  [a]          - Aktion hinter einem File
  [b]          - Aktion vor einem File
  [c]          - keine Warnung beim Erzeugen eines Archivs
  [N]          - use instance [count] of name
  [f]          - Abschneiden von Filenamen
  [P]          - Benutze volle Pfadnamen
  [o]          - erhalte Datum und Uhrzeit
  [u]          - File nur ersetzen, wenn neuer als entsprechendes
                 File im Archiv
```

Sonstiges:

```
  [S]          - Keine Symboltabelle erzeugen
  [v]          - Verbose
  [V]          - Ausgabe der Versionsnummer
```

gdb – GNU Debugger

Der GNU Debugger emöglicht es, in einzelnen Schritten durch ein C-Programm zu laufen und so herauszufinden, an welchen Stellen ein Fehler aufgetreten ist. Der GNU Debugger ist ein Werkzeug für die Programmentwicklung. Allerdings lassen sich auch core-Files mit ihm auswerten und laufende Prozesse beobachten. Das zu testende Programm wird in der Kommandozeile spezifiziert. Die Arbeit mit dem GNU Debugger wird erleichtert, wenn das Programm nicht mit strip bearbeitet wurde und spezielle Optionen bei der Compilation angegeben wurden:

```
    gcc -g [123] ....

gdb [Optionen] <ausführbares-file> <core-file>

gdb [Optionen] <ausführbares-file> <process-id>

gdb [Optionen] <ausführbares-file> [Argumente des Programms ...]
```



```
Optionen:
   -s  file     - Laden der Symboltabelle aus Symbol-File
   -e  file     - debuggen des Programm-Files
   -c  file     - debuggen des Core-Files
   -x  file     - lesen von Debug-Kommandos aus File
   -d  dir      - Quelldateien aus Direktory lesen
   -q           - Quit, leiser Start
   -batch       - nur -x und .gdbinit beachten
   -cd=dir      - Wechseln in Direktory
   -f           - volle Filenamen

gdb-Befehle(1)

   backtrace    - Anzeigen, wie das Programm an die aktuelle Stelle
                  gekommen ist
   breakpoint   - Setzen eines Breakpoints (help breakpoint)
                      break, awatch, disable, enable
   cd           - Wechseln in aktuelles Verzeichnis
   clear        - Löschen des letzten Breakpoints
   commands     - Kommandos für Breakpoint
   continue     - fortsetzen nach Break
   delete       - Löschen eines Breakpoints
   display      - Anzeigen von Variablen
   down         - Abwärts im Stack
   info         - Informationen ausgeben (info breakpoint)
   jump         - im Quelltext springen
```

gdb-Befehle (2)

```
kill        - abbrechen
list        - Quelltexte an der aktuellen Stelle ausgeben
next        - nächste Zeile im Quelltext ausführen
print       - Ausgabe von Variablen
pwd
quit        - beenden von gdb
run         - starten des Programms
search      - suchen im Quelltext
step        - nächste Zeile im Quelltext ausführen
              bei Funktionscall, halt in der ersten Zeile
up          - aufwärts im Stack
watch       - Breakpoint für Datenzugriff
```


renice - Ändern der Priorität eines laufenden Prozesses

```
renice Priorität [[-p] <PIDs> ] [[-g] <PGRPs> ] [[-u] <Benutzernamen> ]
renice --help
```

Mittels renice kann man die Priorität (nice-Wert) eines laufenden Prozesses anzeigen und verändern. Die Angabe von mehreren Prozess-nummern bzw. mehreren Prozessgruppen bzw. von mehreren Benutzernamen ist möglich. Wird ein Benutzername angegeben, werden alle Prozesse des entsprechenen Benutzers in der Priorität verändert. Dies gilt auch für die Prozesse einer Prozessgruppe

```
Linux
-----

strace - trace alle Systemcalls eines Programms bzw. eines Prozesses

strace [-dffhiqrtttTvVxx] [-a column] [-e expr] ... [-o file]
       [-p pid] ... [-s strsize] [-u username] [-E var=val] ...
       [command [arg ...]]

strace -c [-e expr] ... [-O overhead] [-S sortby] [-E var=val] ...
       [command [arg ...]]

    -c           - count time, calls, and errors for each syscall
                   and report summary
    -f           - follow forks, -ff -- with output into separate files
    -F           - attempt to follow vforks, -h -- print help message
    -i           - print instruction pointer at time of syscall
    -q           - suppress messages about attaching, detaching, etc.
    -r           - print relative timestamp
    -t           - print absolute timestamp,
    -tt          - print absolute timestamp with usecs
    -T           - print time spent in each syscall, -V -- print version
    -v           - verbose mode: print unabbreviated argv, stat,
                   termio[s], etc. args
    -x           - print non-ascii strings in hex,
    -xx          - print all strings in hex
    -a column    - alignment COLUMN for printing syscall results
                   (default 40)
```



```
    -e expr      - a qualifying expression: option=[!]all or
                   option=[!]val1[,val2]...
                   options: trace, abbrev, verbose, raw,
                            signal, read, or write
    -o file      - send trace output to FILE instead of stderr
    -O overhead  - set overhead for tracing syscalls to OVERHEAD usecs
    -p pid       - trace process with process id PID, may be repeated
    -s strsize   - limit length of print strings to STRSIZE chars
                   (default 32)
    -S sortby    - sort syscall counts by: time, calls, name,
                   nothing (default time)
    -u username  - run command as username handling setuid
                   and/or setgid
    -E var=val   - put var=val in the environment for command
    -E var       - remove var from the environment for command
```

```
free - zeigt den freien Speicherplatz an

Das  free Kommando gehört zu der ps-Suite.  Wie das ps Programm muss
es bestimmte Daten direkt aus dem  Kernelspeicher  lesen.

free [-b|-k|-m|-g] [-l] [-o] [-t] [-s delay] [-V]

  -b,-k,-m,-g Anzeigen in Bytes, KBytes, MBytes, or GBytes
  -l   Anzeigen ausführlich
  -o   Altes Format anzeigen (keinen -/+buffers/cache Zeile)
  -t   Anzeigen einer  total-Zeile für  RAM + swap
  -s <delay>  Aktualisieren der Anzeige alle <delay> Sekunden
  -V   Anzeigen der Version
```



```
vmstat - Statistik für Virtuellen Speicher ausgeben

Ausgabe von Statistikinformationen über die Nutzung und die
Kapazitäten des virtuellen Speichers einschließlich zugehöriger
E/A-Opterationen und E/A-Bereiche

vmstat [-V]
vmstat [-a] [-n] [Intervall [Anzahl]]
vmstat [-f] [-s] [-m]
vmstat [-S Maßeinheit]
vmstat [-d]
vmstat [-p platten-partition]

    -V prints version.
    -a Ausgabe der aktiven und inaktiven Seiten
    -d Ausgabe der Platten-Statistik
    -D Ausgabe der Platten-Informationen
    -s Ausgabe der Seitenstatistik
    -m Ausgabe Pufferstatistik (slabinfo)
    -S Maßeinheit k, K, g oder G
    --help  dieser Text
```

```
top - Anzeige der Auslastung der CPU

Anzeigen der Auslastung der CPU und anderer Ressourcen (z.B. Speicher
durch die Prozesse. Die Ausgabe kann unter verschiedenen Gesichts-
punkten sortiert werden.

top -hv | -bcHisS -d delay -n iterations -p pid [, pid ...]

   -h, -v   - Hilfstext und Version
   -b       - Batch-Mode
   -d delay - Delay-Zeit
   -i iter  - Wiederholungen

   top-Subkommandos

     h   - Hilfe
     k   - kill
     r   - renice
     l,m,t   - Umschalten Zusammenfassungen: l - load average
                   m - Speicherauslastung   t - cpu-status
     1   - CPU-Wechsel
     M   - Nach Speichernutzung sortieren
     O   - Auswahl des Sortier-Kriteriums
     o   - Auswahl der angezeigten Werte
     c   - Umschalten lange Kommandozeile
     u   - Nutzer auswählen
     z   - Umschalten Color-Modus (Z einstellen)
     b   - Umschalten Bold-Modus  (B einstellen)
```



```
slabtop - Anzeigen der Speichernutzung durch den Kernel

slabtop gibt eine detailierte aktuelle Aufstellung der Benutzung
des Caches des Kernels für die einzelnen Module. Die Ausgabe kann
unter verschieden Gesichtspunkten sortiert werden.
slabtop [Optionen]

Optionen:
  --delay=n, -d n    Anzahl der Sekunden zwischen Aktualisierung
  --once, -o         Nur einmal anzeigen
  --sort=S, -s S     Sortieren nach Sortierkriterium
  --version, -V      Anzeige der Version
  --help             Anzeige Hilfstext

Sortierungskriterien (Sub-Kommandos)
  a: Sortierte Ausgabe nach Anzahl aktiver Objekte
  b: Sortierte Ausgabe nach Objekte pro Slab
  c: Sortierte Ausgabe nach Cache-Size
  l: Sortierte Ausgabe nach Anzahl von Slabs
  v: Sortierte Ausgabe nach Anzahl von aktiven Slabs
  n: Sortierte Ausgabe nach Name
  o: Sortierte Ausgabe nach Anzahl von Objekten
  u: Sortierte Ausgabe nach Auslastung
  p: Sortierte Ausgabe nach Seiten pro Slab
  s: Sortierte Ausgabe nach Objekt-Größe
```

```
Solaris
-------

truss - Protokollieren von allen Systemcalls und Signalen


Mittels  truss  kann man alle Systemcalls eines Prozesses protokollieren.
Der Prozess kann dabei von truss gestartet werden oder bereits laufen.
Die Ausgabe kann sehr umfangreich werden. Lässt sich aber auch genau
eingrenzen.

truss [-fcaeildD] [-[tTvx] [!]<syscalls>] [-[sS] [!]<signals>] \
        [-[mM] [!]<faults>] [-[rw] [!]<fds>] \
        [-[uU] [!]<libs>:[:][!]<funcs>] [-o outfile] \
         command | -p pid[/lwps] ...

Aufrufmöglichkeiten von truss:

   truss  -p pid        - Liste von Prozessen und LWPs,
                          die getracet werden sollen. Die
                          Prozesse laufen schon.

   truss  command       - auszuführendes Kommando mit Optionen,
                          das getracet werden soll
```



```
Optionen:

   -a  - Argumentstring bei exec anzeigen
   -e  - Environment bei exec anzeigen
   -d  - mit Timestamps in jeder Zeile
   -f  - bei fork oder vfork Kindprozess folgen
   -l  - LWP-ID mit einfügen
   -o outfile - Protokollfile
   -r[!]fd - Liste von Files bei denen bei einem read() der vollständige
         Puffer ausgegeben werden soll (all = alle Files) (default: -r!all)
   -t[!]syscalls - Liste von Systemcalls, die beobachtet werden sollen
   -s[!]sig - Liste von zu beobachtenden Signalen (default: -sall
   -u[!]lib,...[:[!]func,...] - Liste von Bibliotheken und Funktionen, die ge
         sollen
   -v[!]syscalls  - Liste von Systemcalls, bei denen alle übergebenen Steuerb
         vollständig ausgegeben werden (default: -v!all)
   -w[!]fd - Liste von Files bei denen bei einem write() der vollständige
             Puffer ausgegeben werden soll (all = alle Files) (default: -w!al
   -x[!] syscalls - Liste von Systemcalls, Ausgabe der Parameter in Hex (defa
   -c  - Statistik

Beispiele:
       truss -o ls.prot  ls -lisa
       truss  -rall -vall -wall -xall aefdl -p 3456
```

```
prstat   - Anzeigen der Aktivitäten von Prozessen oder Nutzern

prstat erzeugt eine Ausgabe der momentanen Aktivitäten von Prozessen
bzw. Nutzern (top). Die Ausgabe kann einmalig, interaktiv oder
mehrmalig erzeugen. Die Ausgabe kann für Nutzer, Prozessen und
Prozessgruppen spezifiziert werden.

prstat [-acJLmRtTvZ] [-u <euidlist>] [-U <uidlist>]
        [-p <pidlist] [-P <cpulist] [-C <psrsetlist>]
        [-j <projidlist] [-k <taskidlist] [-z <zoneidlist>]
        [-s <key> | -S <key>] [-n <nprocs>[,<nusers>]]
        [<interval< [<counter>]]

-a   - Prozesse und Nutzer anzeigen
-c   - Ausgabe nacheinander
-J   - Ausgabe von Prozessen und Projekten
-L   - LWL-Prozesse mit anzeigen
-m   - Microstatus anzeigen
-R   - Scheduling-Informationen mit anzeigen
-s key - Sortieren nach key
       cpu, pri, rss, size, time
-t   - Ausgabe nach Nutzern
```



```
gcore    - Erzeugen eines Core-Images eines aktiven Prozesses

Mittels gcore kann man Core-Images von den spezifizierten
Prozessen erzeugen. Der Name des Core-Images-Files ergibt sich
aus "core." und der Prozess-ID.

gcore [-pgF] [-o filename] [-c content] process-id...

  -c content      Teile des Speichers, die in den Dump sollen:
                  all, anon, ctf, data, dism, heap, ism, rodata, shanon
                  shm, shfile, stack, symtab, text

  -F              Force. Ziehe den Dump immer.

  -g              Dump schreiben in Global-Core-Repository

  -o filename     Filename des Dumpfiles

  -p              Dump entsprechen dumpadm-Spezifikation (root)

  process-id      Prozess-Nummer
```

```
plimit   - Lesen und Setzen der Resourcen-Limits eines
           laufenden Prozesses

Mit plimit kann man die Hard und Soft-Limits eines Prozesses
anzeigen lassen. Ebenfalls ist das Setzen von Hard- und Soft-Limits
für einen laufendne Prozess möglich
Der Prozess wird mittels PID spezifiziert.

Ressourcen-Limits anzeigen:

  plimit [-km] pid...

     -k     File-Size in KB anstellen in 512-Byte Blöcken
     -m     File-Size in MB  (1024*1024)
```



```
Ressourcen-Limits setzen:

  plimit {-cdfnstv} soft,hard... pid...

Mittels plimit kann nur der Super-User oder der Eigentümer
Limits setzen. Nur der Super-User kann Hard-Limits vergrößern.
   -k     File-Size in KB anstellen in 512-Byte Blöcken
   -m     File-Size in MB  (1024*1024)
Jede Option wird durch einen Soft- und einen Hard-Wert
charakterisiert. Folgende Maßeinheiten gibt es:
    nk       n KB
    nm       n MB (Minuten für CPU Zeit)
    nh       n Stunden (für CPU Zeit)
    mm:ss    Minuten und Sekunden (für CPU Zeit)
Soft-Limit kann nicht größer als Hard-Limit sein.
   -c soft,hard    Core File Size Grenzen (Standard
                   512-Byte Blöcke).
   -d soft,hard    Heap size limits
   -f soft,hard    File Size Grenzen (Einheit ist 512-
                   Byte Blöcke).
   -n soft,hard    File Descriptor Grenzen
   -s soft,hard    Stack Segment Größe (Default Einheit Standard KB)
   -t soft,hard    CPU-Zeit Grenze  (Default Einheit ist
                   Sekunden).
   -v soft,hard    Virtuelle Memory Size

   pid      Prozess ID Liste.
```

coreadm - core file administration

coreadm specifies the name and location of core files produced by abnormally-terminating processes. See core(4). Only users who have the sys_admin privilege can execute the first form of the SYNOPSIS. This form configures system-wide core file options, including a global core file name pattern and a core file name pattern for the init(1M) process. All settings are saved in coreadm's configuration file /etc/coreadm.conf to set at boot. See init(1M). Nonprivileged users can execute the second form of the SYNOPSIS. This form specifies the file name pattern and core file content that the operating system uses to generate a per-process core file.

```
coreadm [-g pattern] [-G content] [-i pattern] [-I  content]
        [-d option...] [-e option...]

coreadm [-p pattern] [-P content] [pid...]

coreadm  -u
```


preap - Beenden eines Zombies

Beenden eines defuncten Prozesses (Killen von Zombies). Wenn der Elternprozess nicht auf den Kindprozess wartet, entsteht ein Zombie-Prozess. Mittels preap können diese Zombie-Prozesse entfernt werden und der Eintrag in der Proc-Tabelle freigegeben werden.

```
preap [-F] pid

     -F  - Force
```

```
dtrace   - dynamic tracing compiler and tracing utility


DTrace is a comprehensive dynamic tracing framework for  the
Solaris Operating System. DTrace provides a powerful infras-
tructure that permits administrators, developers,  and  ser-
vice personnel to concisely answer arbitrary questions about
the behavior of the operating system and user programs.

The Solaris Dynamic  Tracing  Guide  describes  how  to  use
DTrace to observe, debug, and tune system behavior. Refer to
this book for a detailed  description  of  DTrace  features,
including the bundled DTrace observability tools, instrumen-
tation providers, and the D programming language.

The dtrace command  provides  a  generic  interface  to  the
essential  services provided by the DTrace facility, includ-
ing:
  o  Options that list  the  set  of  probes  and  providers
     currently published by DTrace

  o  Options that enable probes directly using  any  of  the
     probe  description  specifiers (provider, module, func-
     tion, name)

  o  Options that run the D compiler and compile one or more
     D  program  files  or  programs written directly on the
     command line
```



```
  o  Options that generate anonymous tracing programs

  o  Options that generate program stability reports

  o  Options  that  modify  DTrace  tracing  and  buffering
     behavior and enable additional D compiler features

You can use dtrace to create D scripts by using it in  a  #!
declaration  to create an interpreter file. You can also use
dtrace to attempt to compile D programs and determine  their
properties  without  actually  enabling tracing using the -e
option. See OPTIONS. See the Solaris Dynamic  Tracing  Guide
for  detailed  examples  of how to use the dtrace utility to
perform these tasks.

dtrace [-32 | -64]  [-aACeFGHhlqSvVwZ] [-b bufsz]  [-c  cmd]
       [-D  name  [=value]]  [-I  path]  [-L  path] [-o output] [-s
       script] [-U name] [-x arg [=val]] [-X  a | c | s  |  t]  [-p
       pid]  [-P  provider  [[predicate]  action]]  [-m [provider:]
       module [[predicate] action]] [-f [[provider:] module:]
       function  [[predicate] action]] [-n [[[provider:] module:]
       function:] name [[predicate] action]]
       [-i probe-id  [[predicate] action]]
```

```
pmap     - display information about the address space of a process

The pmap utility prints information about the address  space
of a process.

/usr/bin/pmap [-rslF] [pid | core] ...
/usr/bin/pmap -x [-aslF] [pid | core] ...
/usr/bin/pmap -S [-alF] [pid | core] ...

The pmap utility prints information about the address  space
of a process.
Options
  -a       Prints anonymous and swap reservations  for  shared
              mappings.
  -F       Force. Grabs the target  process  even  if  another
              process has control.
  -l       Shows unresolved dynamic linker map names.
  -r       Prints the process's reserved addresses.
  -s       Prints HAT page size information.
  -S       Displays swap reservation information per  mapping.
  -x       Displays additional information  per  mapping.


./s_sock &
pmap 14637
pmap -x -a 14637
```



```
trapstat - Ausgabe eines Reports über alle  Traps

/usr/sbin/trapstat   [-t | -T | -e entry]
              [- C processor_set_id  |  -c cpulist]
              [-P] [-a] [-r rate] [ [ interval [count]] |
              command | [args]]

/usr/sbin/trapstat -l
```

```
pargs    - Ausgabe der Argumente und Umgebungsvariablen eines
           laufenden Prozesses oder eines core-Files

Mittels pargs erhält man die vollständige Liste der Argumente
bzw. der Umgebungsvariable eines laufenden Prozesses bzw. eines
mit Core-Dump abgestürzten Prozesses.

pargs [-aceFl] [pid | core] ...

Optionen:

   -a   Argumente ausgeben (default).
   -c   Ausgabe als 7-Bit-ASCII
   -e   Umgebungsvariable ausgeben
   -F   Force.
   -l   Ausgabe als jeweils eine Kommandozeile

   pid      Process ID Liste.
   core     Process core file.

./sock asdfasd asf asdf sad sad fas fsd f asd aad d &
pargs  15123    # Argumente
pargs -a 15123  # identisch
pargs -e 15123  # Environment
pargs -l 15123  # Kommandozeile
kill -3 15123
pargs -a core
gdb -c core
gdb s_sock core
```