Einführung Unix/Linux

Kapitel: Grundlagen

Jens Roesen <jens@roesen.org>

Würzburg, März 2010
Version: 0.1.5 – **sehr beta** –

Dieses Dokument wurde in vim (`http://www.vim.org`) bzw. TEXnicCenter (`http://www.texniccenter.org/`) geschrieben und mit LaTeX (`http://www.latex-project.org/`) formatiert und gesetzt.
Die jeweils aktuelle Version ist unter `http://www.roesen.org` erhältlich.

Inhaltsverzeichnis

Vorwort

Motivation

Im Rahmen interner Schulungsmassnahmen kam die Frage nach geeigneten Schukungsmaterialien bzw. einem Skript für Unix-Neulinge auf. Schulungsunterlagen und Skripte für Einsteiger, aber letztendlich hat mir bei den meisten entweder etwas gefehlt, oder es war für unseren Zweck viel zu viel irrelevanter Stoff. Da wir in der Hauptsache mit Solaris und Linux-Systemen arbeiten und dabei Themen wie X-Windows oder Druckerverwaltung komplett ausklammern können, aber auf Themen wie Netzwerke, Troubleshooting, Mailserver oder DNS-Server Wert legen, habe ich mich irgendwann hingesetzt und angefangen dieses Skript zu schreiben.

Es ist der Versuch Systemadministratoren mit Grundkenntnissen in Unix den Arbeitsalltag zu erleichtern und dabei zwei verschiedene Unix-Geschmacksrichtungen, nämlich Solaris[1] und Red Hat Enterprise Linux, gleichermassen zu betrachten.

Mir ist durchaus klar, dass nicht alles im Folgenden beschriebene „state of the art“ ist bzw. sein kann und sicher auch noch etliche Fehler übersehen wurden. Wer einen solchen findet, weiss wie man einige Aufgaben besser lösen kann oder bessere Beispiele kennt ist herzlich eingeladen mir eine Mail an <jens@roesen.org> zu schicken.

Zielgruppe

Dieses Kurzscript ist als Crashkurs zur Einführung in die Administration von Unix und Linux Systemen gedacht. Es wird dabei ausschliesslich auf der Konsole und ohne grafische Oberfläche gearbeitet. Die gezeigten Beispiele beziehen sich auf Systeme unter Sun Solaris und RedHat Enterprise Linux.

Ohne Vorkenntnisse und Erfahrung mit Unix und/oder Linux Systemen wird der angesprochene Stoff teils nur schwer zu verstehen sein. Als alleiniges Lehrskript für blutige Anfänger ist es daher nicht geeignet obwohl in einigen Kapiteln vereinzelt kurz auf Grundlagen eingegangen wird (z.B. Kapitel 2).

Aufbau des Skripts

Wirr. Durch und durch. Aber zu mehr ist momentan keine Zeit. Ich habe versucht die Kapitel und Themen in eine halbwegs sinnvolle Reihenfolge zu bringen. Im Lauf der Zeit wird da sicherlich noch einiges umgestellt werden.

[1] In der vorliegenden Version des Skripts nur bis Version 9.

Typographisches

Da sich alle Beispiele, Kommandos und Ausgaben auf der Konsole abspielen, werden diese Bereiche entsprechend formatiert um sich vom regulären Text abzusetzen. Nach dem Login als User root, mit dem wir in diesem Skript hauptsächlich arbeiten werden, landet man in einem Command Prompt der so aussehen koennte:

```
[root@server1 root]#
```

Da dieser Prompt ja nach name des Systems oder aktuellem Verzeichnis mal kürzer aber auch sehr viel länger sein kann, wird der root Prompt auf

```
#
```

verkuerzt. Bitte den Hash (`#`) hier **nicht** als Kommentarcharakter verstehen, der unter Linux/Unix z.B. in Shellskripten die folgende Zeile vor der Ausführung durch die Shell schützt. Der normale User-Prompt, falls er uns wirklich einmal begegnen sollte, wird analog dazu auf

```
$
```

zusammengestrichen.

Für Konsolenausgaben, Konfigurationsdateien oder Teile von Skripten wird eine nichtproportionale Schrift verwendet:

```
if [ -n "$_INIT_NET_IF" -a "$_INIT_NET_STRATEGY" = "dhcp" ]; then
        /sbin/dhcpagent -a
fi
```

Werden in einem Beispiel Konsoleneingaben vom Benutzer erwartet, wird die in einer nichtproportionale Schrift dargestellt, wobei die Benutzereingaben fett gedruckt sind:

```
# uname -a
SunOS zoidberg 5.9 Generic_118558-21 sun4u sparc SUNW,Sun-Blade-100
```

Kommandos, Dateinamen oder Benutzerkennungen im laufenden Text werden ebenfalls in einer nichtproportionalen Schrift dargestellt: „Mit dem Befehl `pwd` kann überprüft werden, in welchem Verzeichnis man sich gerade befindet.“.

Müssen in einem Beispiel noch Teile der erwarteten Benutzereingaben durch die richtigen Werte ersetzt werden, so wird dieser Teil in kursiver Nichproportionalschrift dargestellt: „Für jedes Interface welches beim boot konfiguriert werden soll muß eine Datei **/etc/hostname.*interface*** existieren.“.

Eigennamen, Personen oder Organisationen erscheinen manchmal (ich bin gerade zu faul alle Vorkommen entsprechend zu formatieren) in Kapitälchen: „Eine sehr große Rolle hierbei hat die University of California in Berkley (UCB) gespielt, an der Thompson im Winter 76/77 eine Vorlesung zum Thema Unix abhielt.“.

1 Unix Grundlagen

> Unix is user-friendly. It just isn't promiscuous about which users it's friendly with.
>
> *(Stephen King)*

1.1 Login am System

Da es sich bei Unix-Systemen um Multiuser-Systeme handelt, muß man sich in aller Regel zuerst am System anmelden bevor man mit ihm arbeiten kann. Dazu benötigt man z.B. einen Benutzernamen um sich gegenüber dem System zu authentifizieren und ein beispielsweise ein Passwort, ein Zertifikat oder einen Key für eine erfolgreiche Authorisierung.

Anmelden kann man sich entweder lokal am System oder, wie im Serverbetrieb eher üblich, „remote“ über eine Netzwerverbindung. Je nach Anmeldeart können verschiedene Authentifizierungsmethoden eingesetzt werden. In der Regel greift man über eine SSH Verbindung mit Passwort oder Public-Key Authentifizierung auf ein System zu. Eine Remoteanmeldung per Telnet sollte heutzutage in jedem Fall vermieden werden, da Telnet im Klartext übertragen wird und somit Passwörter etc. leicht mitgelesen werden können.

Nach dem Login wird die für den Benutzer vorgegebene Shell gestartet und man landet automatisch im eigenen Homedirectory, in dem sich einige für den Login relevante Daten befinden und ausserdem alle persönlichen Daten gespeichert werden. Welche Shell beim Login gestartet wird und welches das Homedir ist, wird in der Datei `/etc/passwd` festgelegt. Dazu in einem späteren Kapitel eventuell mehr :P.

Die Shell stellt die Schnittstelle zwischen dem Benutzer und dem Kernel, dem Betriebssystemkern, dar. Sie nimmt alle Befehle entgegen, leitet sie entsprechend weiter und gibt uns über verschidene „Kanäle“ Rückmeldungen. Auf all das werden wir im Kapitel „Arbeiten auf der Shell“ genauer eingehen.

1.2 Befehle und Kommandos

Befehle unter Unix bestehen oft nicht nur aus einem einzelnen Wort, dem Befehl oder Programmnamen an sich, sondern auch aus zusätzlichen Optionen und Argumenten. In Abbildung 1.1 auf Seite 2 habe ich einmal einen typischen Befehl inklusive Prompt in die einzelnen Teile zerlegt. `tar` ist das Programm welches gestartet wird, die Optionen

`-xzfv` sagen `tar` wie es mit dem Argument `irssi_0.8.5.tar.gz`, in diesem Fall einer Datei, verfahren soll.

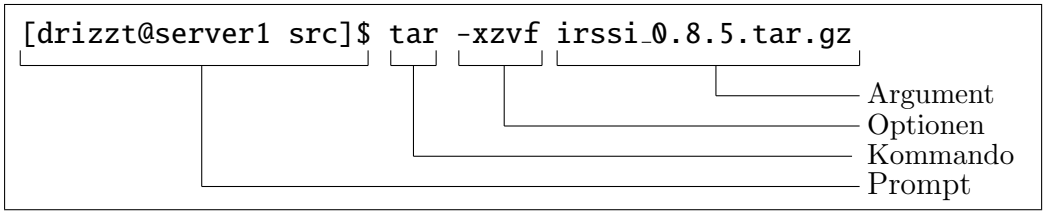

Abbildung 1.1: Unix Befehlssyntax

Viele der im täglichen Betrieb verwendeten Befehle sind sogenannte „shell builtins“. Diese Befehle sind in die Shell integriert und es werden keine externen Befehle, also irgendwo im Filesystem befindlichen ausführbare Dateien, gestartet. Shell builtins arbeiten schneller als externe Befehle, da sie direkt in die Shell integriert sind und nicht erst geladen werden müssen. Der Nachteil an builtins ist, das sie nicht einzeln modifiziert oder upgedatet werden können. Falls das nötig sein sollte, muss die komplette Shell ersetzt werden. Beispiele für builtins sind `cd`, `pwd`, `logout` oder `echo`. Hilfe zu den builtins bekommt man mittels `help <builtin>`:

```
# help pwd
pwd: pwd [-LP]
    Print the current working directory.  With the -P option, pwd prints
    the physical directory, without any symbolic links; the -L option
    makes pwd follow symbolic links.
```

Die meisten builtins stehen zusätzlich auch als externe Programme zur Verfügung (z.B. `/bin/pwd`). Eine grosse Ausnahme stellt hierbei der Befehl `cd` zum wechseln von Verzeichnissen dar, der nur als builtin existiert. Ob man es mit builtins oder externen Befehlen zu tun hat kann man mit `type` testen:

```
# type pwd
pwd is a shell builtin
# type tar
tar is /bin/tar
```

1.3 Erste Schritte im System

Da wir eingeloggt sind nun grob wissen wie Befehle aufgebaut sind können wir die ersten Schritte unternehmen. Dazu werde ich einige der grundlegenderen Befehle vorstellen mit denen man täglich arbeitet. Ich werden die Befehle nicht bis ins allerkleinste Detail beschreiben. Weitere Optionen und Hilfen erhält man bei fast allen Befehlen über *befehl* `--help` oder über die entsprechende Manpage die mit `man <befehl>` aufgerufen wird. Mit den Manpages beschäftigen wir uns in Abschnitt 1.4 noch genauer.

1.3.1 pwd - aktuelles Verzeichnis abfragen

`pwd` gibt den absoluten Pfad des aktuellen Arbeitsverzeichnisses an. Nicht mehr und nicht weniger:

```
# pwd
/usr/local/bin
```

1.3.2 ls - Verzeichnisinhalte anzeigen lassen

Um sich den Inhalt eines Verzeichnisses anzeigen zu lassen benutzt man das Kommando `ls`. Dadurch werden alle Datei und Unterverzeichnisnamen im aktuellen Arbeitsverzeichnis angezeigt, deren Namen nicht mit einem '`.`' beginnen.

```
# ls
ausfuehrbar      symlink  testdir          testfile
```

Um auch versteckte Dateien, solche mit einem `.` am Anfang vom Dateinamen, anzeigen zu lassen, wird die Option `-a` benutzt. Dabei steht '`.`' für das aktuelle und '`..`' für das übergeordnete Verzeichnis.

```
# ls -a
.                ..               .versteckt       ausfuehrbar      symlink
testdir          testfile
```

Um sich erweiterte Informationen (Dazu mehr in Kapitel ??) anzeigen zu lassen wird die Option `-l` verwendet. `-l` wird auch sehr gerne mit `-a` kombiniert.

```
# ls -la
total 3314
drwxr-xr-x   3 drizzt   drizzt       512 Sep  5 16:38 .
drwx------  44 drizzt   drizzt      4608 Sep  5 15:28 ..
-rw-r--r--   1 drizzt   drizzt    671744 Sep  5 16:38 .versteckt
-rwxr--r--   1 drizzt   drizzt    999424 Sep  5 16:38 ausfuehrbar
lrwxrwxrwx   1 drizzt   drizzt        12 Sep  2 13:04 symlink -> /home/drizzt
drwxr-xr-x   3 drizzt   drizzt       512 Sep  5 16:35 testdir
-rw-r--r--   1 drizzt   drizzt        17 Aug 24 22:40 testfile
```

1.3.3 cd - Verzeichnisse wechseln

Um in ein anderes Verzeichnis zu wechseln verwendet man den Befehl `cd <`*`zielverzeichnis`*`>`. Wird `cd` ohne Argument verwendet, wechselt man automatisch in sein Homedir.

```
# pwd
/usr/local/bin
# cd
# pwd
/root
```

Gibt man als Argument den Namen des Verzeichnisses an, in da es zu wechseln gilt, muß zwischen *relativen* und *absoluten* Pfadnamen unterscheiden. Ein absoluter Pfad beginnt mit einem `/` — es wird also von der Wurzel, root, des Verzeichnis-/Dateibaums ausgegangen. Relative Pfade beginnen nicht mit einem `/`. In diesem Fall wird vom aktuellen Arbeitsverzeichnis ausgegangen.

```
# pwd
/
# cd /home/drizzt/
# pwd
/home/drizzt
# cd Unix/testdir/
# pwd
/home/drizzt/Unix/testdir
```

Wird als Shell die Bash verwendet kann man mit `cd -` in das Verzeichnis zurückwechseln, in dem man sich vor dem letzten Verzeichniswechsel befand. Welche Shell man gerade verwendet kann man durch den Befehl `echo $0` anzeigen lassen[1]

```
# echo $0
-bash
# pwd
/root
# cd /usr/local/bin
# pwd
/usr/local/bin
# cd -
/root
# pwd
/root
```

1.3.4 whoami und who - wer bin ich und wer ist sonst noch da

Falls man sich mal nicht so sicher sein sollte wer man selber ist (kommt nach stundenlangem Getippe durchaus vor) kann man sich vom System mit `whoami` (Linux) bzw. `who am i` (Linux und Solaris) helfen lassen.

```
$ whoami
drizzt
$ who am i
drizzt   pts/6        2008-11-11 15:18 (192.168.1.70)
```

Wer ausser einem noch auf dem System unterwegs ist, wird durch den Befehl `who` angezeigt.

```
$ who
vt100 tty1         2008-10-29 14:13
vt100 pts/2        2008-10-29 14:14 (:0:S.1)
vt100 pts/3        2008-10-29 14:14 (:0:S.2)
drizzt   pts/6        2008-11-11 15:18 (192.168.1.70)
```

[1] In der C-Shell funktioniert dies leider nicht interaktiv, wohl aber über ein Shellscript. Interaktiv wird die Fehlermeldung `No file for $0` ausgegeben.

1.3.5 date - Datum und Uhrzeit

Mit dem Befehl `date` kann man sich die aktuelle Uhrzeit und das Datum anzeigen lassen.

```
# date
Thu Sep  5 10:41:43 MEST 2004
```

Die Ausgabe der Datums und der Uhrzeit kann auf vielerlei Arten formatiert werden[2]

```
# date '+%d.%m.%y %H:%M:%S'
05.09.04 10:43:29
```

Der User `root` kann mittels `date` die Systemzeit manuell setzen. Eine genaue Uhrzeit ist kein Luxus sondern später beim Troubleshooting ein sehr wichtiges Hilfsmittel. Dabei müssen auch eventuell unterschiedliche Zeitzonen[3] berücksichtigt werden.

1.3.6 cat - Dateiinhalte anzeigen lassen

Der Befehl `cat` liest eine Datei ein und gibt deren Inhalt auf dem Monitor wieder. Der Schalter `-n` gibt zusätzlich noch die Zeilennummern aus, während `-b` die Ausgabe der Zeilennummern bei Leerzeilen unterdrückt:

```
# cat testfile
Zeile 1

Zeile 3
# cat -n testfile
     1  Zeile 1
     2
     3  Zeile 3
# cat -b testfile
     1  Zeile 1

     2  Zeile 3
```

1.3.7 less & more - Dateiinhalte seitenweise anzeigen lassen

Wenn man sich lange Textdateien mittels `cat` anzeigen läßt rauscht die Ausgabe förmlich an einem vorbei. Man kann in diesem Fall die Datei mittels `more`[4] (`more <dateiname>`) oder `less`[5] (`less <dateiname>`) seitenweise ausgeben lassen.

Es gibt gibt verschiedene Möglichkeiten innerhalb von mit `more` und `less` angezeigten Dateien zu navigieren oder zu suchen. In Tabelle 1.1 sind die wichtigsten gegenübergestellt. Grade bei `less` gibt es oft mehrere Möglichkeiten ein und dasselbe auszuführen. In diesen Fällen hab ich mich auf die gängigsten Methoden beschränkt.

Wie man sieht ist `less` um einiges mächtiger als `more` und versteht oft auch Kommandos die wir beim Arbeiten mit dem Editor `vi` noch kennenlernen werden.

[2] Die Formatierungsmöglichkeiten sind in der Manpage von `date` genauer beschrieben.
[3] Siehe auch `http://de.wikipedia.org/wiki/Zeitzone` oder `http://www.zeitzonen.de/`
[4] `more` ist normalerweise auf jedem Unix System verfügbar.
[5] `less` gehört leider immer noch nicht bei jedem System zum Standard-Befehlsrepertiore

Befehle innerhalb von `more` und `less`		
Auswirkung	`more`	`less`
Eine Bildschirmseite runter	`SPACE z`	`SPACE f`
Eine Bildschirmseite rauf	`b`	`b`
Eine Zeile runter	`RETURN`	`RETURN e`
Eine Zeile rauf		`y k`
Eine halbe Seite weiter		`d`
Eine halbe Seite zurück		`u`
Hilfeseite aufrufen	`h`	`h H`
Abwärts nach `pattern` suchen	`/pattern`	`/pattern`
Aufwärts nach `pattern` suchen		`?pattern`
Zur nächsten Fundstelle von `pattern` weiter unten im Text springen	`n`	`n`
Zur nächsten Fundstelle von `pattern` weiter oben im Text springen		`N`
Suchmarkierungen löschen		`ESC-u`
An den Anfang springen		`g`
Ans Ende springen		`G`
Zur Zeile *n* springen	`:n`	`:n pn`
Datei `file` anzeigen		`e: file`
Datei in `$EDITOR` bearbeiten	`v`	`v`
Beenden	`q`	`q :q Q :Q ZZ`

Tabelle 1.1: Befehle innerhalb von `more` und `less`

1.3.8 `passwd` - eigenes Passwort wechseln

Passwörter sollten regelmäßig gewechselt werden und möglichst sicher sein. Unter Unix wird das eigene Passwort mit dem Befehl `passwd` geändert. Wenn man als normaler, unprivilegierter User `passwd` aufruft muss man zunächst sein aktuelles und danach zwei mal das neue Passwort eingeben. Als Superuser root entfällt der erste Schritt und mann kann auch mit `passwd <username>` die Passwörter anderer User ändern.

Bitte bei der Auswahl eines Passworts wirklich darauf achten, dass es sicher ist. Sicher bedeutet mindestens acht Zeichen lang, Gross- und Kleinbuchstaben sowie Ziffern und Sonderzeichen sollten enthalten sein. Am einfachsten nimmt man sich einen Satz und Kombiniert die Anfangsbuchstaben der einzelnen Worte zu einem Passwort. Aus „Ein sicheres Passwort besteht aus mindestens 8 Zeichen!“ wird so das Passwort „EsPbam8Z!“.

1.3.9 `logout` - Ausloggen

Aus einem System ausloggen kann man sich mit dem Befehl `logout` oder der Tastenkombination `CTRL-D` (`^D`). Genauer gesagt beendet man damit die aktuelle Shell. Wenn

man also aus der `sh` heraus erst noch die `bash` gestartet hat muss man sich zwei mal ausloggen um das System letztendlich zu verlassen.

1.4 Manpages, die schnell verfügbare Hilfe

Normalerweise sind auf jedem Unix System auch die Manpages installiert. Dabei handelt es sich um eine Sammlung von etlichen Dokumenten mit oft sehr genauer und detailierter Dokumentation zu Befehlen oder Systemdateien. Die Manpages sind in momentan 12 Bereiche unterteilt.

Zum anzeigen dieser „online Dokumentation“ benutzt man den Befehl `man`. Leider empfinden viele User das Lesen der Manpages als nervig, langwierig oder gar zu kompliziert. Man findet jedoch kurzfristig — und oft auch langfristig — keine besseren Anleitungen.

Es existiert zu nahezu jedem Unix Kommando bzw. Systembestandteil eine Manpage. Man ruft sie auf, indem man einfach `man` gefolgt von dem Kommandonamen aufruft, zu dem man Hilfe braucht. Abbildung 1.2 auf Seite 9 zeit die erste Bildschirmseite von `man man`.

Die Navigation innerhalb der Manpage ist der Navigation innerhalb von `more` und `less` sehr ähnlich, da die Manpage in `more` bzw. `less` gepiped wird. Durch ein „`-`“ als Option wird `man` angewiesen sein Ausgabe in `cat` zu pipen und die Manpage rauscht in einem Schwung durch. Mit `h` kann man sich in `man` eine Hilfeseite mit Befehlen anzeigen lassen auf der man auch prompt mit `@(#)more.help 1.5` bzw. `SUMMARY OF LESS COMMANDS` begrüßt wird.

Mit `e` oder `RETURN` scrollt man eine Zeile weiter runter, mit `y` eine Zeile weiter rauf. Ganze Seiten runter geht es mit `f` oder `SPACE` und wieder rauf geht's mit `b`. Nach `pattern` sucht man mit `/pattern` wobei mit `n` zum nächsten Fundort von `pattern` weiter runter gesprungen wird und mit `N` zum nächsten Fundort weiter oben in der Manpage.

Wenn das System keine Manpage mit dem angegebenen Befehl findet, kann man innerhalb der Manpages nach Begriffen (`keyword`) suchen. Dazu muessen die Manpages vom Superuser indiziert werden. Das geschieht mittels `catman` bzw. `mandb` oder `makewhatis`. Anschliessend kann man sich mit `whatis` eine einzeilige Zusammenfassung zum Suchbegriff ausgeben lassen.

```
# whatis hostname
hostname        hostname (1)    - set or print name of current host system
```

Eine erweiterte Suche bietet `apropos`. Damit werden die Manpages mit zugehöriger Sektion ausgegeben, bei denen `keyword` entweder im `NAME` oder im `DESCRIPTION` Bereich der Manpage vorkommt.

```
# apropos hostname
check-hostname  check-hostname (1m) - check if sendmail can determine the system's
                                      fully-qualified host name
ethers          ethers (4)      - Ethernet address to hostname database or domain
gethostname     gethostname (3c)    - get or set name of current host
```

```
gethostname     gethostname (3xnet) - get name of current host
hostname        hostname (1)    - set or print name of current host system
```

```
MAN(1)                          Manual pager utils                          MAN(1)

NAME
       man - an interface to the on-line reference manuals

SYNOPSIS
       man  [-c|-w|-tZ]  [-H[browser]]  [-T[device]] [-X[dpi]] [-adhu7V] [-i|-I]
       [-m system[,...]] [-L locale] [-p string] [-C file] [-M path] [-P pager]
       [-r prompt]  [-S list] [-e extension] [--warnings [warnings]] [[section]
       page ...] ...
       man  -l [-7] [-tZ] [-H[browser]] [-T[device]] [-X[dpi]] [-p string]
       [-P pager] [-r prompt] [--warnings[warnings]] file ...
       man -k [apropos options] regexp ...
       man -f [whatis options] page ...

DESCRIPTION
       man  is the systemâs manual pager. Each page argument given to man is
       normally the name of a program, utility or function.  The manual page
       associated  with each of these arguments is then found and displayed.
       A section, if provided, will direct man to look only in that  section
       of  the manual.  The default action is to search in all of the avail
       able sections, following a pre-defined order and  to  show  only  the
       first page found, even if page exists in several sections.

       The  table  below shows the section numbers of the manual followed by
       the types of pages they contain.

       1   Executable programs or shell commands
       2   System calls (functions provided by the kernel)
       3   Library calls (functions within program libraries)
       4   Special files (usually found in /dev)
       5   File formats and conventions eg /etc/passwd
       6   Games
       7   Miscellaneous (including macro  packages  and  convenâ[m
           tions), e.g. man(7), groff(7)
       8   System administration commands (usually only for root)
       9   Kernel routines [Non standard]

       A manual page consists of several parts.

       They  may  be  labelled  NAME, SYNOPSIS, DESCRIPTION, OPTIONS, FILES,
       SEE ALSO, BUGS, and AUTHOR.
 [...]
```

Abbildung 1.2: Die man-Manpage