

## General

**cal:** prints calender  
**hostname:** name of computer  
**id:** UID (userid), GID (group)  
**ps:** current processes  
**uname:** name of operating system  
**tty:** terminal device used  
**who:** all users logged in  
**whoami:** current user  
**man [command]:** help manuel  
**less [filename]:** print file page per page  
**which [comm]:** show program path  
**bc:** binary calculator

## Wildcards

\* -- beliebige Zeichenfolge  
? -- genau 1 Zeichen  
a[abc] -- aa, ab, ac  
a[!abc] -- ad, ae... a[jedes ASCI außer abc]  
\ escape - "ab?cd" -> "ab?cd"  
mkdir home/{dir1,dir2,dir3} -- erstellt 3 dirs

## Textfiles

**cut:** Spalten/Felder, nummeriert ab 1,  
Trennzeichen: Standard: Tab, spez. mit -d  
-f: Spalten auswählen  
-c: chars auswählen  
**cut** -f2,3 tex.txt | **cut** -c 1-3  
**tr** -s " " -- tr: translate --- s: squeeze -- hier: ersetzt mehrere Leerzeichen zu einem  
**sort** -- aufsteigend -r --absteigend -n --numerisch (sonst lexikographisch) -t --Trennzeichen -k2 -- zweite Spalte untersuchen  
**uniq:** Gleicher Inhalt als 1 Zeile -c -- gefundene Anzahl  
**head / tail:** default: ersten / letzten 10 Zeilen  
-nl -- erste/letzte n Zeilen: -5l --ersten 5 = -n5  
**wc:** word count:  
-l -- Anz. Zeilen, -c -- Anz. Bytes, -L -- Anz. Zeichen längste Zeile  
-w -- Anz Wörter

## Rechteverwaltung

-neue Verzeichnisse / files: **umask** 022 //danach werden neue Verzeichnisse mit 755 und files mit 644 erstellt  
**-SUID/SGID:** s statt x bei user: ausführender Benutzer hat während Ausführung die UID / GID des Besitzers, Bsp.:  
passwd  
**-SUID** bei Verzeichnis macht keinen Sinn  
**-SGID** bei VZ erbt diese Gruppe - sinnvoll für Teamverzeichnisse  
**-Sticky-Bit:** bei VZ kann jeder mit Schreibrecht Daten löschen oder umbenennen. - Lösung: Sticky-Bit: verhindert das. nur root und owner können löschen und umbenennen  
**-chmod** -- erstes bit x000 legt fest. StickyBit: 1; SGID: 2; SUID:4

## Git diff

Befehl	Änderungen von	→	auf
git diff	Staging Area		Arbeitskopie
git diff --cached	HEAD		Staging Area
git diff HEAD	HEAD		Arbeitskopie
git diff HEAD^ HEAD	Vorgänger von HEAD		HEAD
git diff branch1 main	branch1		main
git diff branch1...main	Ursprung von branch1		main

## Filemanagement

**ls:**  
-a: hidden files  
-l longlist, mit Rechte  
-d \*/ directories in current only  
-R list recursively

**mkdir** -p --Unterverzeichnisse autom. erstellen  
**rmdir** dirnam --del emty dir  
**rm** -d --dir -r -rekursiv

**mv, cp**  
mv f1 f2 dir1 -- f1 & f2 in dir1 schieben  
mv -i --fragt ob überschreiben  
mv f1 dir1/fnew -- f1 in dir 1 schieben und umbenennen  
**cp** file1 file2 -- kopieren und umbenennen  
cp -R dir1 dir2 -- dir1 rekursiv kopieren  
cp -i file1 dir/fileA -- file1 kopieren und umbenennen -- ggf nachfragen (-i)

## Benutzerverwaltung

-mit Schreibrecht auf Verzeichnis kann man darin löschen / umbenennen  
**umask:** Rechte enziehen: umask 022 = entzieht G und o das Writerecht  
-Gruppe **sudo:** alle darin dürfen sudo aufrufen  
-sudo **adduser** (--system) username (--ingroup groupname)  
-sudo **deluser** (--remove-home) username  
-sudo **usermod** -aG groupname username //add user to group  
-sudo usermod -g groupname username //change primary group  
-sudo deluser username groupname //rem. user from group  
-sudo **addgroup** groupname  
-sudo **delgroup** groupname  
-sudo usermod -l newname oldname //rename user - wenn user ausgelogged  
-sudo **passwd** username  
**root:** standardmäßig in Ubuntu deaktiviert - passwort setzen  
- dann: sudo passwd -l (lock) -u (unlock) root

-**chmod** -R g-x,u+wx,o=r filename

-chmod 1755 -- stickybit -- chmod +t

-chmod 2755/3755/4755..

## Eigentümer / Gruppe ändern

-**chown** username filename // nur root  
-**chgrp** groupname filename // root oder owner wenn in Gruppe  
-chown user:group files //beliebig setzen möglich  
-chown -R root /home //rekursiv

## GIT Setup

- git **init**: neuer Workspace
- git **clone**: kopiert bestehendes repo --> neuer workspace
  - o benennt das Ursprungsrepo --> wichtig für syncs
  - b auszucheckender branch
- git **config -l** --vorhandene sets
  - unset --entfernt setting
  - system (all repo aller users) --global (alle repos des users)
  - local (akt. repo, default)
- **Alias** setzen: git config --global alias.newcommand "orig. command"
- git config --global **user.name**/ user.email "user name"/"email"

## Datenstrukturen:

- **Commit** speichert: Stand von Arbeitsverzeichnis: Metadaten (wer, wann etc.), **parent** ID, tree ID (hashwerte)
- **tree** speichert: hashes von Blobs
- **blob**: Abbild einer ganzen Datei - wenn ein Hash einer zu committenden Datei schon existiert, wird einfach der bereits erstellte Blob verwendet, ansonsten wird ein neuer Blob erstellt

## Zustände: untracked, unmodified, modified, staged

- git **rm** file: löscht und added, also = rm file; git add file
- git **mv** file file1 = mv ..; git add file; add file1: Git erkennt Verschiebungen durch den gleichen Hashwert
- git **status**: Zustand der files im WD
- git **diff** (--cached) --vergleicht WD mit Index bzw. Index mit repo (--cached)
- **.gitignore**: \*bel. string; \*\* bel. Unterverzeichnisse; ? bel. char; [abc]:ein Zeichen der Menge; mit !file wird td getrackt
- git **commit**: -m -a(alles vom index) --amend(zum letzten commit dazu)

## Commits referenzieren

- ^ und ~: beides Vorgänger. HEAD~2 = HEAD~~ / ^^
- HEAD^^2 = 2 commits zurück, beim 2. aber anderer branch
- auch main^ geht
- git **log**: Historie
- git **show**: infos zu commit: z.B. git show HEAD^^ oder hash

## Rückgängig machen

- git **diff**: Änderungen zwischen commits
- git **checkout**: bewegt HEAD und neues WD
- git **reset** --soft (ändert nur head) --mixed (default, ändert staging) --hard (ändert WD)
- git **restore** -S (--staged; repo to staging) -W (--worktree; staging to WD)
- git **revert** (erzeugt neuen commit, inhalt gleich altem Commit)
- bei geteilten repos immer revert
- git **reflog**: alte Aktionen

## Files Zusammenführen

- git **branch** //neuer branch
- git **checkout** // Head -- branch/commit // -b legt branch an
- git **switch** -c (legt branch an) -d (detach head auf beliebigen comm)
- git **merge** feature (Quellbranch) (Zielbranch ist der ausgewählt)
- git **rebase** (neue Basis) -i (interactive)
- git **cherry-pick** (-n no commit)
- git **stash**: Zwischenspeicher in einem Stack

## Remote repositories

- git **clone**: lokale Kopie von entferntem repo
  - Name ist standardmäßig origin, kann durch -o <name> geändert werden
  - beim clone werden alle branches runtergeladen und als **remote tracking branch** gespeichert, automatisch wird der branch ausgecheckt auf den der head im remote repo zeigt
  - Beim auschecken eines solchen wird ein **local tracking branch** erstellt, auf dem man arbeiten kanneinfach git switch -c <branchname>  
git switch -c <newname> origin/<remotename>

- git **remote** -v (list entfernte repos) **show** (details zu ent. repo) **add** (ent. repo hinzufügen) **remove** (ent. repo. entf.)
- git **fetch**: aktualisiert rem. track. branches, noch nicht gemerged
- git **pull**: fetch und merge
- git **push**: -u (set upstream (macht zu local tracking branch)) -d (del, löscht branch im rem. repo) -f (**force push**)

Leere Verzeichnisse können nicht versioniert werden!  
Deshalb brauchen wir darin eine versteckte Datei z.B. gitkeep, um sie zu versionieren.

## Docker

- docker **ps**: processes running?
- docker **pull** {image}
- docker **images**: local avail. images
- docker **run -it -rm** (remove when ended) {image} (creates new container)
- docker **container ps -a** (list all containers)
- docker container **inspect** {container} (config file)
- docker **start / stop** {container}
- docker **attach** {container} (attaches terminal to PID1 process)
- docker **exec** -it {container} bash (starts proc bash, not directly PID1)
  - exit only termin. the bash -- more safe

## persisting data

- docker **volume ls** (list volumes) - used to save outside of container layer
- docker run -it --mount source=wse-volume, destination=/wse-data/ ubuntu
- docker run -it -v wse-volume:/wse-data(:ro (read only)) ubuntu

## Bind mounting:

- docker run -it --rm -v \${PWD}/data:/wse-data ubuntu
  - mounts a directory from the host system

