Introduction to Linux: **System Tools**

Philipp Fruck & Pius Walter

System Tools

Goal of this lecture

- Better understanding of your local system
- Learn how to search files and text efficiently
- Understand tools for inspecting system state and resource usage
- Get familiar with archiving, compression, and basic networking commands

Why?

- These tools are essential for troubleshooting and system administration
- They build directly on the fundamentals from the previous lecture

Utilities

Command	Description	Useful Flags / Examples
`man`	Show **man**ual pages	`man ls`, `man hier`
`tldr`	Shorter command descriptions (**t**oo **l**ong **d**idn't **r**ead)	`tldr ls`
`alias`	Define shortcuts	`alias ll="ls -l"`
`unalias`	Removes defined shortcuts	`unalias ll`
`command`	Executes a command without aliases, etc.	`command ls`
`watch`	Repeat a command periodically	`watch -n .5 ls -l` `-c` to add **c**olor

Utilities

Command	Description	Useful Flags / Examples
`man`	Show **man**ual pages	`man ls`, `man hier`
`tldr`	Shorter command descriptions (**t**oo **l**ong **d**idn't **r**ead)	`tldr ls`
`alias`	Define shortcuts	`alias ll="ls -l"`
`unalias`	Removes defined shortcuts	`unalias ll`
`command`	Executes a command without aliases, etc.	`command ls`
`watch`	Repeat a command periodically	`watch -n .5 ls -l` `-c` to add **c**olor

> alias only applies to the current shell session. We will learn how to persist it later.

Utilities

Command	Description	Useful Flags / Examples
man	Show **man**ual pages	man ls, man hier
tldr	Shorter command descriptions (**t**oo **l**ong **d**idn't **r**ead)	tldr ls
alias	Define shortcuts	alias ll="ls -l"
unalias	Removes defined shortcuts	unalias ll
command	Executes a command without aliases, etc.	command ls
watch	Repeat a command periodically	watch -n .5 ls -l -c to add **c**olor

> alias only applies to the current shell session. We will learn how to persist it later.

> watch cannot resolve aliases
> To fix: alias watch="watch " or watch -n .5 bash -ic ll

Finding Files & Text

Command	Description	Useful Flags / Examples
`find`	Search for files	`find /etc -type f -name "*.conf"`
`locate`	Search files via prebuilt index	`updatedb` updates index
`grep`	Search inside files	`-i` ignore case, `-r` recursive, `-n` show line numbers, `-v` invert match
`ripgrep`	Recursive `grep` on steroids :)	`rg -i searchterm`
`fzf`	Interactive fuzzy search	

System Info

Command	Description	Useful Flags / Examples
`uname`	Show system info	`-a` all details
`uptime`	Show uptime & load average	
`lsblk`	List block devices	`--fs` shows file system type
`mount`	Mount a filesystem	`mount /dev/sda1 /mnt`
`umount`	Unmount a filesystem	`umount /dev/sda1`, `umount /mnt`
`df`	Show disk usage	`-h` human-readable sizes
`du`	Show space used by files/dirs	`-sh` summarize directory size
`free`	Show memory usage	`-h` human-readable
`ps`	Show process informations	`ps aux` lists all running processes
`top` / `htop`	Show running processes	Interactive tools

System Info

Command	Description	Useful Flags / Examples
`uname`	Show system info	`-a` all details
`uptime`	Show uptime & load average	
`lsblk`	List block devices	`--fs` shows file system type
`mount`	Mount a filesystem	`mount /dev/sda1 /mnt`
`umount`	Unmount a filesystem	`umount /dev/sda1`, `umount /mnt`
`df`	Show disk usage	`-h` human-readable sizes
`du`	Show space used by files/dirs	`-sh` summarize directory size
`free`	Show memory usage	`-h` human-readable
`ps`	Show process informations	`ps aux` lists all running processes
`top` / `htop`	Show running processes	Interactive tools

> btop and bottom (btm) are advanced terminal task managers

> "Mission Center" is a great GUI app!

> 100 % CPU utilization = 1 CPU core fully utilized -> 4-core CPU can have 400 % CPU load

Archiving

On Linux, archives are created using `tar`

- **C**reate an archive and write it to a **f**ile:
  - `tar -cf file.tar file1 dir1/ dir2/`
- Lis**t** the contents of a tar **f**ile **v**erbosely:
  - `tar -tvf path/to/source.tar`
- **C**reate a g**z**ipped archive **v**erbosely and write to **f**ile
  - `tar -czvf file.tar.gz dir/`
- E**x**tract an archive **f**ile into the target directory:
  - `tar -xf file.tar --directory output/`
- E**x**tract a g**z**ipped **f**ile **v**erbosely
  - `tar -xzfv file.tar.gz`

Archiving

On Linux, archives are created using `tar`

- **C**reate an archive and write it to a **f**ile:
  - `tar -cf file.tar file1 dir1/ dir2/`
- Lis**t** the contents of a tar **f**ile **v**erbosely:
  - `tar -tvf path/to/source.tar`
- **C**reate a g**z**ipped archive **v**erbosely and write to **f**ile
  - `tar -czvf file.tar.gz dir/`
- E**x**tract an archive **f**ile into the target directory:
  - `tar -xf file.tar --directory output/`
- E**x**tract a g**z**ipped **f**ile **v**erbosely
  - `tar -xzfv file.tar.gz`

For tar `-` is optional

`tar xzf` works too

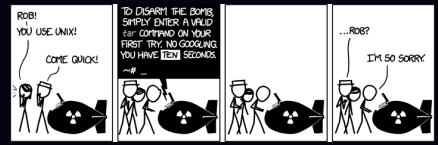
Archiving

On Linux, archives are created using `tar`

- **C**reate an archive and write it to a **f**ile:
  - `tar -cf file.tar file1 dir1/ dir2/`
- Lis**t** the contents of a tar **f**ile **v**erbosely:
  - `tar -tvf path/to/source.tar`
- **C**reate a g**z**ipped archive **v**erbosely and write to **f**ile
  - `tar -czvf file.tar.gz dir/`
- E**x**tract an archive **f**ile into the target directory:
  - `tar -xf file.tar --directory output/`
- E**x**tract a g**z**ipped **f**ile **v**erbosely
  - `tar -xzfv file.tar.gz`

For tar `-` is optional

`tar xzf` works too



https://xkcd.com/1168/

Compression

Command	Compression of	Useful Flags / Examples
`gzip` / `gunzip`	Single file	`gzip --keep /path/to/file` `gunzip --keep archive.gz`
`bzip2` / `bunzip2`	Single file	`bzip2 --keep /path/to/file` `bunzip2 --keep archive.bz2`
`xz` / `unxz`	Single file	`xz --keep /path/to/file` `unxz --keep archive.xz`
`zip` / `unzip`	Files and directories	`zip --recurse-paths my.zip dir1/ dir2/` `unzip archive.zip`

Compression

Command	Compression of	Useful Flags / Examples
`gzip` / `gunzip`	Single file	`gzip --keep /path/to/file` `gunzip --keep archive.gz`
`bzip2` / `bunzip2`	Single file	`bzip2 --keep /path/to/file` `bunzip2 --keep archive.bz2`
`xz` / `unxz`	Single file	`xz --keep /path/to/file` `unxz --keep archive.xz`
`zip` / `unzip`	Files and directories	`zip --recurse-paths my.zip dir1/ dir2/` `unzip archive.zip`

Unix philosophy: One tool should only do one thing

- e.g. `tar` archives, `gzip` compresses, `gunzip` decompresses
- `tar` has `-z` for `gzip`, `gzip` has `--decompress` for `gunzip`
- `zip` comes from the Windows world -> Does archiving + compression
  - Unix filesystem attributes **can** be supported (but don't have to).

Networking Basics

Command	Description	Useful Flags / Examples
`curl`	Transfer data from URLs	`-I` headers only, `-L` follow redirect `-X` change HTTP method, `-o` **o**utput file
`wget`	Download files from web	`-r` **r**ecursive download, `-o` **o**utput file
`ping`	Test connectivity	`ping -c 4 dhbw.de`: **c**ount 4 replies
`mtr` / `traceroute`	Print route packets trace	`mtr dhbw.de`
`ss` / `netstat`	Show network connections	`ss -tulpn` show listening ports `-t` = TCP, `-u` = UDP
`dig` / `nslookup`	DNS queries	`dig A example.com`

Exercise

To navigate the filesystem more efficiently we want to install a terminal-based file manager (manually).

Task: Install `lf` (https://github.com/gokcehan/lf) within the terminal.

- Download latest `lf-linux-amd64.tar.gz` via `wget`/`curl` from the GitHub releases
- Extract the archive
- Copy `lf` binary to your users `PATH`
  - Choose a path in the home directory
  - Create the directory if necessary
- Run `lf` to navigate your home directory

Time: 10 minutes

SSH: Secure Remote Access

Lets you connect to a remote system

- Stands for **s**ecure **sh**ell
- Encrypted remote shell & file transfer
- Works on all major platforms
  - You can use Windows / macOS as well
- Uses client-server model
  - Server: `sshd`
  - Client: `ssh`

SSH: Secure Remote Access

Lets you connect to a remote system

- Stands for **s**ecure **sh**ell
- Encrypted remote shell & file transfer
- Works on all major platforms
  - You can use Windows / macOS as well
- Uses client-server model
  - Server: `sshd`
  - Client: `ssh`

- SSH client stores fingerprints of the remote system
  - Upon first connection you will be prompted to accept the remote fingerprint
  - Type "yes" if you trust the connection (verify the fingerprint)
  - If the fingerprint changes (e.g. the system has been reinstalled or compromised), SSH will refuse future connection

SSH: Secure Remote Access

There are multiple authentication methods:

- username/password: "Weak" authentication (password strength matters)
- SSH key/certificate based: "Strong" authentication via cryptographic methods
  - Running `ssh-keygen -t ed25519` will create an ssh key pair
    - Optional (recommended): Set a password to store it encrypted
    - `~/.ssh/id_ed25519`: Private key - don't share with anyone
    - `~/.ssh/id_ed25519.pub`: Public key - upload via moodle

> If you run the command a second time, make sure to choose a different name for the SSH key. Don't override your existing key. Choose a path inside ~/.ssh/ for now.

SSH: Secure Remote Access

There are multiple authentication methods:

- username/password: "Weak" authentication (password strength matters)
- SSH key/certificate based: "Strong" authentication via cryptographic methods
  - Running `ssh-keygen -t ed25519` will create an ssh key pair
    - Optional (recommended): Set a password to store it encrypted
    - `~/.ssh/id_ed25519`: Private key - don't share with anyone
    - `~/.ssh/id_ed25519.pub`: Public key - upload via moodle

> If you run the command a second time, make sure to choose a different name for the SSH key. Don't override your existing key. Choose a path inside ~/.ssh/ for now.

Basic usage

```
ssh <user>@<host>
```

e.g. `ssh s123456@dhbw`

SSH: Secure Remote Access

There are multiple authentication methods:

- username/password: "Weak" authentication (password strength matters)
- SSH key/certificate based: "Strong" authentication via cryptographic methods
  - Running `ssh-keygen -t ed25519` will create an ssh key pair
    - Optional (recommended): Set a password to store it encrypted
    - `~/.ssh/id_ed25519`: Private key - don't share with anyone
    - `~/.ssh/id_ed25519.pub`: Public key - upload via moodle

> If you run the command a second time, make sure to choose a different name for the SSH key. Don't override your existing key. Choose a path inside ~/.ssh/ for now.

Basic usage

```
ssh <user>@<host>
```

e.g. `ssh s123456@dhbw`

Common options

```
# custom port, default is 22
ssh -p 2222 <user>@<host>
# use specific key ("identity")
ssh -i ~/.ssh/id_ed25519 <host>
```

Thank you!

Thank you for your attention!

Don't forget the feedback in Moodle please!