



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386



Tools 2026 – Computing @ ZITI

Shigehiko Schamoni
23.04.2026

Overview

1 Compute Resources

- URZ
 - Services
- ZITI
 - Compute Cluster

2 Compute Systems

- Operating Systems
- Linux

3 Accessing Compute Resources

- Secure Shell
- Kerberos Authentication
- Batch System

Compute Resources for ZITI Students

- General URZ-provided services
(heiBOX, heiCONF, heiCHAT, etc.)
- ZITI Resources
 - PC Pool in U012 / OMZ
 - Student PCs in room 543, INF 368
 - Research group specific infrastructure
 - Compute server
 - Compute cluster
- BaWü Cluster
 - Registration required
 - BWUniCluster
 - BWForCluster

URZ-Resources for Students

- heiBOX
 - Limited size but quite useful and safe
 - Access for non-university members possible
- heiCHAT
 - Matrix-based chat, requires UniID
 - Android app: Element
- heiCONF
 - Virtual classroom, based on BigBlueButton (open source)
- PC Pool
 - URZ building INF 330 / Room E11 (ground floor)
 - Monday to Friday, 8:00 – 16:00
 - MS Windows PCs with standard software (MS Office, Adobe Acrobat, etc.)
- <https://www.urz.uni-heidelberg.de/en/support/it-for-all-target-groups/it-for-students>

Windows Terminal Server at URZ

- MS Windows PCs for remote access
 - Standard software installed (MS Office, Adobe Acrobat, etc.)
- Accessible from the University network (eduroam/VPN) only
 - From MS Windows:
 - Remote Desktop Connection App
 - From MacOS:
 - Windows App
 - From Linux:
 - Multiple clients, e.g. Remmina, xfreerdp3
- More Information
`https://www.urz.uni-heidelberg.de/de/service-katalog/desktop-und-arbeitsplatz/windows-terminalserver`

ZITI Research Groups

APPLICATION SPECIFIC COMPUTING

Application Specific Computing

Prof. Dr. Robert Strzodka



Computer Architecture Team

Prof. Dr. Nima TaheriNejad



Hardware & Artificial Intelligence

Prof. Dr. Holger Fröning



Neuromorphic Computing Architectures

Prof. Dr. Johannes Schemmel



NOVEL COMPUTING TECHNOLOGIES

Novel Computing Technologies

Prof. Dr. Dirk Koch



Circuit Design Group

Prof. Dr. Peter Fischer



Figure: <https://www.ziti.uni-heidelberg.de/en/research/>

Hierarchical Networks

- The university network is a subnet of the internet
- The ZITI network is a subnet of the university network
- ZITI research groups often have their own subnets within the ZITI network
- Multiple levels of security:
 - Firewalls between networks
 - User access restrictions
 - IP-based access restrictions

ZITI Infrastructure

- PC Pool in U012 / OMZ
- Hardware Lab in U013 / OMZ
- Student's room 543 in INF 368, 5th floor
 - Three ZITI administered desktop computers
 - Several displays, mice, keyboards, docking stations
- Three server rooms in INF 368 on different floors
 - 40 GBit floor switches
 - 10 GBit room switches
 - ZITI internal resources
 - Compute Server
 - Compute Cluster
- Additional labs for experimental setups, e.g.
 - ASC: Mellanox interconnect cluster
 - HAWAII: embedded lab, benchmark lab
 - ...

Accessing ZITI Infrastructure

- Central resources (PC pool, student's PCs)
 - No registration required
 - Simply use you UniID
- Research group specific resources
 - Administered by group admin
 - Requires certain group membership to log in
 - Compute server
 - Compute cluster
 - Lab infrastructure
- University-wide resources
 - Usually accessible via UniID
 - BW clusters need registration

Why to Use a Compute Cluster

- Processes require more resources than a single server or a workstation provides
- Parallel processing of large amounts of data
- Huge models that require distributed computation
- Tuning of model parameters
- Typical properties of a cluster:
 - Sharing of resources
 - Uses a batch system such that different user's tasks don't interfere
 - Load of RAM, CPU cores, GPU resources

Clusters for ZITI

- ZITI's research groups
 - HAWAII-Cluster
 - ASC-Cluster
 - NCA-Cluster
- bwUniCluster
 - HPC system for general purpose and teaching
 - <https://wiki.bwhpc.de/e/BwUniCluster2.0>
 - <https://wiki.bwhpc.de/e/BwUniCluster3.0>
- bwForCluster
 - HPC system for research
 - <https://wiki.bwhpc.de/e/Registration/bwForCluster>
- Common to all clusters above:
 - Slurm batch system
 - <https://slurm.schedmd.com/documentation.html>

Server Operating Systems

- MS Windows
 - Windows Server is an industry standard
 - URZ uses Active Directory, Exchange, etc.
 - ZITI has no Windows Server instance
- MacOS
 - Apple discontinued macOS Server on April 21, 2022
 - Apple silicon (M-chips) has become interesting for ML/AI tasks
- Linux / Unix
 - The most reasonable system for compute intensive tasks

Linux Introduction

- Linux is a derivative of a Unix operating system
- Other Unix systems:
 - BSD
 - Solaris, IBM AIX, SINIX
 - macOS X(!)
- Components:
 - Kernel: provides system calls, file systems, a networking stack, etc.
 - Set of optional programs, e.g.
 - Shell to execute commands
 - Graphical window subsystem
 - Compilers, runtime environments, editors
 - ...

Linux Distributions

- A Linux distribution compiles a Linux kernel and different applications
- Different distributions have different use cases
 - Headless server operation
 - Desktop optimized
 - Embedded systems
 - Gaming platform
- Different focuses, e.g.
 - Stability of applications and services
 - Bleeding-edge software
- Some common distributions
 - Debian
 - Ubuntu (based on Debian), Mint (based on Ubuntu)
 - Fedora, Red Hat
 - Arch Linux
 - DSL, ...

Working with Linux (I)

- Linux is a multi-user system $\Rightarrow$ inherently supports *multiple users*
- Central component to do work: the *terminal* / *shell*
- Classic terminal:
 - Hardware: a terminal device to send and receive commands
 - No computing power, just a display and a keyboard
 - Connected to a mainframe computer somewhere
- Terminal nowadays:
 - A program providing a command prompt
 - Possibly in a graphical window
- Shell:
 - Software: processes input from a user and returns an output
 - Collection of basic built-in commands and control structures
 - Different shell variations: `sh`, `bash`, `zsh`, ...

Working with Linux (II)

- All modern Linux distributions work with different window managers
 - GNOME
 - KDE
 - Cinnamon (MINT)
 - MATE (lighter)
 - XFCE (lightest)
 - Tiling window managers (e.g. Awesome, Hyprland)
- Window managers implement a display server protocol
 - X Window System (X11)
 - Wayland Compositors
- X Window System
 - Slower than Wayland
 - Allows remote X connections

File System in Linux

- If you have been using MS Windows
 - Drives (partitions) are represented by letters (C:\, D:\, ...)
 - Path separator is the \ (backslash)
 - There is no *global* file root
- In Linux / Unix everything is organized in a hierarchical way
 - Paths are relative or absolute referencing the *file root*
 - The root is represented by a preceding “slash character” /
 - Path separator is the slash /
 - Paths can be specified relative to the user’s home represented by a tilde ~
- True hierarchical namespace
 - Every object must implement the file API
 - *Everything* is a file!

File Types

```
@mc1:/opt0/emacs$ ls -al
total 273
drwxr-xr-x  4 install staff    9 Nov 10 2006 .
drwxr-xr-x 23 root   root    31 Oct  6 15:02 ..
-rw-r--r--  1 install staff  7424 Aug 15 2006 basis-utils.el
drwxr-xr-x  2 install staff    36 Aug 15 2006 csmode
-rw-r--r--  1 install staff  5886 Aug 15 2006 doxygen.el
-rw-r--r--  1 install staff 200322 Aug 15 2006 folding.el
-rw-r--r--  1 install staff 15242 Sep  8 2008 rdl-mode.el
drwxr-xr-x  2 install staff    3 Aug 15 2006 skel
-rw-r--r--  1 install staff 277360 Aug 15 2006 verilog-mode.el
```

The command `ls` lists the contents of the current directory.

`ls -l` adds detailed information, e.g. file type, permissions, ...

- `-` : regular files
- `d` : directories
- `b` : block files, e.g. SSDs or HDDs
- `c` : character files: special input devices, e.g. terminal or mouse
- `p` : pipes for inter-process communication
- `s` : sockets for inter-process communication
- `l` : links to another file object

File Permissions

```
@mc1:/opt0/emacs$ ls -al
total 273
drwxr-xr-x  4 install staff    9 Nov 10 2006 .
drwxr-xr-x 23 root    root    31 Oct  6 15:02 ..
-rw-r--r--  1 install staff   7424 Aug 15 2006 basis-utils.el
drwxr-xr-x  2 install staff    36 Aug 15 2006 csmode
-rw-r--r--  1 install staff   5886 Aug 15 2006 doxygen.el
-rw-r--r--  1 install staff 200322 Aug 15 2006 folding.el
-rw-r--r--  1 install staff  15242 Sep  8 2008 rdl-mode.el
drwxr-xr-x  2 install staff     3 Aug 15 2006 skel
-rw-r--r--  1 install staff 277360 Aug 15 2006 verilog-mode.el
```

`ls -la` lists detailed information on all files (even hidden files)

- Every file or directory has an owning user and a group
- Permissions are divided into three segments
 - 1 User
 - 2 Group
 - 3 Other (people where user or group do not match)
- Every segment has three permission attributes
 - r : read permission, code 4
 - w : write permission, code 2
 - x : execute permission, code 1

Permissions Codes

- Reminder: Read (4), Write (2), Execute (1)
- Codes are used as bit set. Examples:
 - 1 Permission of 0: Nothing allowed
 - 2 Permission of 5: Read + Execute
 - 3 Permission of 6: Read + Write
 - 4 Permission of 7: Read + Write + Execute
- Permission string follows file type in `ls -l`:
`-rwxr--r--` corresponds to code 744
- Permissions can be changed via the `chmod` command
 - `chmod 755 myfile.txt` : sets permission to 755
(`-rwxr-xr-x`)
 - `chmod o-x myfile.txt` : removes execute permission from other
 - `chmod u+w myfile.txt` : adds write permission to user

Ownership

```
@mc1:/opt0/emacs$ ls -al
total 273
drwxr-xr-x  4 install staff    9 Nov 10  2006 .
drwxr-xr-x 23 root    root    31 Oct  6 15:02 ..
-rw-r--r--  1 install staff   7424 Aug 15  2006 basis-utils.el
drwxr-xr-x  2 install staff    36 Aug 15  2006 csmode
-rw-r--r--  1 install staff   5886 Aug 15  2006 doxygen.el
-rw-r--r--  1 install staff 200322 Aug 15  2006 folding.el
-rw-r--r--  1 install staff  15242 Sep  8  2008 rdl-mode.el
drwxr-xr-x  2 install staff     3 Aug 15  2006 skel
-rw-r--r--  1 install staff 277360 Aug 15  2006 verilog-mode.el
```

- Ownership of files and directories
 - `chown xy123:urz myfile.txt` : changes ownership to user xy123 and group urz
 - To give ownership to another user, you have to be root
 - To give ownership to another group, you need to be member of that group
- The command `groups` lists all groups you are member of

Navigating the directory tree

- In each directory there are two special files that cannot be removed
 - 1 `.` : represents this directory
 - 2 `..` : represents the parent directory of this directory
- Most important commands to browse through files and directories
 - `cd` : change directory
 - `ls` : list directory contents
- Commands to analyze file system usage
 - `du` : directory usage
 - `df` : disk free
- Getting help
 - `<command> -?` : prints out a short usage description
 - `man <command>` : manual page for the command

Commands contd.

■ More useful commands

- `pwd` : print working directory
- `mkdir` : create a new directory
- `rm` : remove files and directories
- `tree` : get a directory tree
- `touch file` : update last modified timestamp of a file
- `cp source destination` : copy a file
- `cat file` : write out file
- `less file` : print out contents of file, quit with q
- `head -N file` : write out first N lines of file
- `tail -N file` : write out last N lines of file
- `wc file` : count lines and characters of file

■ Advanced commands

- `find` : looking for files with specific attributes
- `grep` : global regular expression print
- `sed` : stream editor
- `awk` : complex scripting language for text processing

Linking

- Links are “pseudo-files” that point to another file
- Useful in many situations, e.g.:
 - In case the same file is needed at multiple locations
 - Link always points to the most recent version of a library
 - Linked file consumes minimal additional storage space (meta data only)
- Two types of links
 - Hard link
 - Create with

```
:~$ ln sourcefile targetfile
```
 - Hard-linked files are not distinguishable
 - Very hard to keep track of
 - Can only exist within a single file system (e.g. not across two partitions)
 - Soft link
 - Create with

```
:~$ ln -s sourcefile targetfile
```
 - When the source file is deleted, its links point to no file
 - Can span multiple file systems

Multitasking

- You can start a command to run in the background – and reuse your shell immediately
 - A command is started in the background by adding an ampersand &, e.g.

```
:~$ gedit myfile.txt &
```
 - List jobs in the background: `jobs`
 - Re-gain control of the jobs listed: `fg 1` (for the first job)
 - Send to background again `Ctrl+Z`, followed by `bg`
- Task hierarchy
 - In Linux all processes are launched from a parent process
 - If a parent process terminates, all child processes also terminate
- Identifying processes
 - `htop` : gives a nice overview of the system's resources
 - If `htop` is not installed, try `top`
 - `ps` : report snapshot of current processes
- Terminating a process
 - `kill PID` : terminates process number PID

Exercise: Linux

Login via SSH

- Secure shell (SSH) is the most common way to connect to remote machines.
- Linux, Mac, and BSD users have the command `ssh` installed by default.
- Windows users can use WSL or PuTTY¹
- The command `ssh` expects a username and a hostname separated by an @-sign:

```
:~$ ssh <user>@zitigate.ziti.uni-heidelberg.de
```

`<user>` is typically your UniID.
- The example above creates an ssh-connection to a gate server which enables access to ZITI's compute resources.
- The gate's system is very restricted and should only be used as a *jump host*.

¹<https://www.chiark.greenend.org.uk/~sgtatham/putty/>

Login via SSH contd.

- To jump to a different server, run `ssh` on the gate:

```
:~$ ssh <user>@<hostname>
```

If `hostname` is part of the ZITI domain, you don't need the fully qualified domain name (FQDN).

- The jump host configuration can be automated by editing the client's config file `$HOME/.ssh/config`:

```
Host zitigate
HostName zitigate.ziti.uni-heidelberg.de
User <username>
Host headnode
HostName csg-headnode
User <username>
ProxyJump zitigate
```

- Now one can directly connect to `csg-headnode` by

```
:~$ ssh headnode
```

- Note: multiple failed logins lock out your current IP!

SSH Public Key Authentication

- Avoids being locked out by unsuccessful logins
- Password-less login
- Useful for jumping across multiple machines
- Required:
 - 1 Key pair (public and secret key)
 - 2 Public key listed in `$HOME/.ssh/authorized_keys`
- VERY recommended!

Kerberos Authentication

- Kerberos is a ticket-based network authentication protocol
- Single-sign-on
- Tickets have a limited lifetime
- ZITI network often uses Kerberos
- Most basic commands:
 - 1 `kinit <user>`
Generates a Kerberos ticket for <user>
 - 2 `klist`
Shows Kerberos ticket(s) on current machine
 - 3 `kdestroy`
Removes own ticket(s) on current machine

Exercise: SSH

Slurm

- Users don't have direct access to cluster's compute nodes.
- All computations need to go through the Slurm batch system.
- The batch system collects information on all jobs and distributes them among the nodes.
- There exist several queues (partitions) with different properties:
 - Compute resources (CPU cores, GPUs)
 - File systems
 - Installed software
 - Maximum job's run time
 - etc.
- If the job exceeds its limits it is killed by the Slurm system.

Basic Slurm Commands (I)

- The `sinfo` command returns general information about all accessible queues

```
:~$ sinfo
```

- Jobs are started by using the `srun` command:

```
:~$ srun [-p <partition >] [-w <node >] --pty  
-- <command>
```

If the requested resources are currently not available, the job is queued.

- The `squeue` command shows the status of all active queues

```
:~$ squeue
```

Basic Slurm Commands (II)

- Additional resources can be requested via command line parameters, e.g.

- Allocate a single GPU for a job:

```
:~$ srun --gres=gpu:1 <...>
```

- Allocate a eight CPUs for a job:

```
:~$ srun --cpus-per-task 8 <...>
```

- To cancel a running job, look up its job ID:

```
:~$ squeue -u <username>
```

- Then issue the `scancel` command

```
:~$ scancel <jobid>
```

Note: you can only cancel your own jobs unless you have superuser privileges.

Advanced Slurm Usage

- Slurm supports special shell scripts using an sbatch mechanism
- Command line parameters and shell commands are combined in a single file:

```
#!/bin/bash
#SBATCH --job-name=my_batch_job
#SBATCH --output=my_result.txt
#SBATCH --mail-user=UniID@stud.uni-heidelberg.de
#SBATCH --mail-type=ALL
#SBATCH --partition=compute

# JOB STEPS
srun hostname # example job step
srun echo $CUDA_VISIBLE_DEVICES # another example job step
```

- Handy for starting multiple jobs with different parameters
- Good introduction: <https://hpc.nmsu.edu/discovery/slurm/slurm-commands/>