HPS
https://hps.vi4io.org

Jonathan Decker

Linux Crash Course

Surviving the Shell

Learning Objectives

- Connect to GWDG machines via SSH and access the command line interface
- Navigate the operating system on Linux using the Bash shell
- Edit files using Nano text editor
- Utilize Git for version control of directory trees
- Solve routine tasks by formulating commands and combining existing programs

Preface

- Focus on most important commands
- Additional content for advanced users
- Use this slide deck as lookup during course
- Available for download on course page:
  https://hps.vi4io.org/teaching/summer_term_2022/pchpc
- ← Red box marks a command that you want to remember
- Other commands are nice to know
- Presentation accompanied by exercises
- Breakout room: https://meet.gwdg.de/b/jul-ffv-ljs-7u5

What is a Shell?

- A shell is a command line interpreter
- It takes commands entered via the keyboard to start programs
- **Bash** is the most widespread shell
- A **terminal** is an input/output environment for shells
- The mouse can still be used to select text for copy and paste
- The shell is only an interface through which other programs are started
- A shell can only show textual output

Open a shell:

- **Windows:** WIN + r, type `powershell` and press enter
- **MacOS:** Search for **Terminal** and open it

Table of contents

SSH Client

Windows 10/11:

- Search for **Powershell**, right click, run as administrator
- `Get-WindowsCapability -Online|Where-Object Name -like '*SSH*'`
  If SSH client is not installed run the following command:
  `Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0`
- Confirm that it works with `ssh -V`

MacOS/Linux:

- Search for **Terminal** and open it
- Check your ssh version `ssh -V`

Connecting with SSH

- Place the SSH key you received per mail in your user folder
- **NN** is the number in the key file name
- In PowerShell or Terminal type the following command

```
ssh -i hpctrainingNN hpctrainingNN@login-mdc.hpc.gwdg.de
 -o ProxyCommand='ssh -W %h:%p hpctrainingNN@login.gwdg.de
 -i hpctrainingNN'
```

- Confirm the connection and enter the SSH keys passphrase **twice**
- The passphrase is on the preparation sheet you received
- If you are already in the GÖNET, you only need the first line

Shell Shortcuts Basics

- TAB Auto-complete file/directory names and commands
- TAB + TAB Show all possibilities
- CTRL + c Abort current running process
- ARROW UP/DOWN Cycle through command history
- `clear` Clear screen
- `exit` Close current shell session

Shell Shortcuts Advanced

- CTRL + a Jump to line start
- CTRL + e Jump to line end
- ALT + f Jump forward one word
- ALT + b Jump backward one word
- CTRL + u Cut line to clipboard from start up to cursor
- CTRL + k Cut line to clipboard from end to cursor
- CTRL + w Cut word before cursor to clipboard
- CTRL + y Paste from clipboard
- CTRL + s / CTRL + q Stop/Resume output to screen from running process without stopping the process

Folder Navigation

- `pwd` Print current directory
- `ls` List files and folders in current directory
- `ls -a` Also list hidden files and folders (start with . marks as hidden)
- `ls -la` List all files and folders in long table format
- `ls -a DIR` List all files and folders in target directory
- `cd DIR` Change directory to target directory
- `cd ~` Change to HOME directory
- `cd ..` Change to parent folder
  - `~` Refers to your HOME folder
  - `.` Refers to the current folder
  - `..` Refers to parent of current folder
- A path including spaces `cd "path/with spaces/"` needs to be put in quotes
- `" "` allow for variable expansion, `''` do not

Getting help with a command

- `COMMAND --help`, `COMMAND -h` or `COMMAND help` commonly shows usage options
- `man COMMAND` Opens the manual for a command
  - Mouse wheel for scrolling
  - d / w For scrolling down/up
  - Mouse wheel sometimes does not work via SSH
  - q For quitting the manual
  - Try `man man`
- `whatis COMMAND` See what pages are available
- `man SECTION COMMAND` Open a specific page for a command
- Search for documentation and guides on the internet

File and Folder Permissions

- Files and folders each belong to a user (owner) and a group
- Read, write and execute permission can be set for owner, group and others
- `ls -l` shows these permissions

d	rwx	- - -	- - -	2	decker15	UMIN	1	Mar 31 14:42	test
-	- rw	- - -	- - -	1	decker15	UMIN	16533	Mar 31 14:41	test.txt
type	user perm	group perm	other perm	# of links	owner	group	size	last modified	name

- Type **d** means directory, **-** means file
- Permission **-** means its not set, **r**, **w**, **x** means read, write or execute permission set
- **s** or **S** means, when executing, use owner/group permission
- **T** means files in folder can only be deleted by their owners

Modifying Permission

- `chmod` Command for changing permission
- `chmod (u|g|o|a)(+|-|=)(r||w||x|| ) TARGET`
- `chmod a+r test.txt` Gives everyone read permission
- `chmod g= test.txt` Removes all permission for group
- `chmod u+x test` Allows execution of test
- `chmod -R g+rw test-dir` Makes test-dir and files and folders in it group readable and writable, **-R** flag makes it recursive
- `chmod +t test-dir` Adds sticky-bit **T** to test-dir
- `chmod u+s test` Use owner permission when executing test

Changing ownership

- `chown NEW_OWNER TARGET` Change the ownership of target
- `chgrp NEW_GROUP TARGET` Change the group of target
- The admin or super-user on Linux systems is called root
- `sudo COMMAND` (super-user do) Execute command as admin
- `whoami` Show own username
- `who` Show logged in users
- `w` More information active users

Nano Basic Usage

- **Nano** is a text editor for the terminal
  - Relatively easy to use
  - Alternatives: **emacs**, **vi**, ...
  - Use your preferred editor
- `nano FILE` To start editing, if file does not exist, its created
- Navigate with ARROW-keys and type to edit
- CTRL + o To save as...
- CTRL + s To save (HPC machines have old nano, use CTRL + o instead)
- CTRL + x To exit

Nano Shortcuts 1/2

- ESC Can be used instead of ALT
- CTRL + w Open search
- ALT + w Continue search
- CTRL + w, CTRL + R Open search and replace
- CTRL + c Cancel command
- ALT + a Set mark for selection
- ALT + 6 Copy selected text (area between mark and cursor) to clipboard
- CTRL + k Cut current line or selected text to clipboard
- CTRL + u Paste clipboard at cursor

Nano Shortcuts 2/2

- ALT + u/e Undo/Redo
- CTRL + a/e Jump to line start/end
- CTRL + y/v Scroll page up/down
- CTRL + g Open help window
- CTRL + o Save as..
- CTRL + c Show cursor position
- CTRL + 7 Jump to line number
- ALT + o Enable/Disable conversion of tabs to spaces

Environmental Variables

- Values can be stored in environmental variables
- Some are used for configurations
- `echo $HOME` To see the value of HOME
- `echo -e ${PATH//:/:\\n}` To get a nice output for PATH
- `printenv` or `set` to see all current env vars
- `export NAME=Value` Set variable, no spaces before or after =
- `unset NAME` Unset variable
- Env vars are bound to your session and do not persist after session ends

Persistent settings

- When you login into a Bash shell, it reads `.bash_profile`
- When you open another Bash shell without login, it reads `.bashrc`
- `nano .bash_profile` Open bash profile and make it load .bashrc
- Add this line to it `[[ -f ~/.bashrc ]] && . ~/.bashrc` and save
- `nano .bashrc` To start editing
- Add `export HELLO=hi`
- `alias` Can be used to set command aliases
- Add `alias ll='ls -la'` and save
- `source .bashrc` To load the changes now

Custom Prompt

- By setting the env var `PS1` you can customize your prompt
- Try `export PS1='[\t] \u@\h:\w$'`
- `\t` Gives the current time
- `\u` Gives your username
- `\h` Gives the hostname
- `\w` Gives the current folder
- Search for **bash ps1 generator** on the internet

Create, Copy, Move, Delete

- `touch FILE` Update modification time of file or create empty file
- `rm -i FILE` Delete file with confirmation, confirm with `y`
- `mkdir DIR` Create directory
- `rmdir DIR` Delete directory
- `rm -rf DIR` Delete everything in folder (sub-folders, files, ...)
  use with **great care**, there is **no undo**
- `cp SRC DEST` Copy a file from source to destination
- `cp -R SRC DEST` Copy folders including sub-folders
- `mv SRC DEST` Move a file or folder, also functions as rename

Disk Usage

- `ls -lh` List directory with human-readable sizes
- `du -h DIR` Show size of target folder and sub-folders
- `du -hd1` Do not show size of sub-folders
- `stat TARGET` Show details including size of file oder folder
- `df -hl` Show filesystem usage, look for filesystem mounted on **/**
- `tree` Show tree representation of sub-folders

Read and Search Files

- `cat FILE` Print file content to shell
- `less FILE` Show file content with pager
- `find PATH -name '*.txt'` Find all txt files in path
- `grep PATTERN FILE` Search for pattern in file
- `grep -R PATTERN PATH` Search for pattern in all files in path
- `locate NAME` Find files containing NAME in their filename
- `head FILE` Show first 10 lines of file
- `tail FILE` Show last 10 lines of file
- `diff FILE1 FILE2` Compare files and list differences

Processes

- `top` or `htop` Show current resource usage by processes
  Use htop over top, close with `q` or `CTRL` + `c`
- `ps` List all processes on current shell session
- `ps -u USER` List all processes by a specific user, try `ps -u root`
- `ps aux` or `ps -ef` List all processes by all users
- `kill PID` Stop process with process id
- `COMMAND1 && COMMAND2` Lets you chain multiple commands
  this will execute COMMAND1 and then COMMAND2
  but only if COMMAND1 succeeded

Jobs

- `COMMAND &` Let the command execute as a background job
- CTRL + z Stop and make the running command a background job
- `jobs` List your background jobs
- Jobs are bound to your shell session, all jobs are killed when you close your shell
- `bg %JOB_NUM` Start a stopped background job
- `fg %JOB_NUM` Move a job into the foreground
- `disown %JOB_NUM` Disown a job from your shell, keeps it running after closing shell

Gain Information on Host System

- `hostname` Show hostname of system
- `uname -a` Show kernel information
- `cat /etc/os-release`, `hostnamectl`, `lsb_release -a`
  Show kernel and distribution information
- `uptime` Show system uptime, time since last restart

Redirect Command Outputs

- `COMMAND > FILE` Redirects the output of command into file
- `>` Creates or overwrites file, `»` creates or appends file
- `|` A pipe that forwards inputs from one command into another
- `ps aux | grep PATTERN` Filter the output of a command using grep
- `COMMAND | sort -u` Sort and filter unique lines in output
- Only the output of the last command is shown in the shell

Bash History

- `history` List all previous commands
- `history -c` Clear history (in case you entered your password)
- `history | grep PATTERN` Look for a command you used before
- `!N` Expands to line n of your bash history
- `!!` Expands to previous command
- `!TEXT` Expands to last command starting with text
- `!?TEXT` Expands to last command containing text
- `!#:N` Expands to nth argument of current command, can be used like this:
  - `mkdir NEW_DIR && cd !#:1` to create and switch to new dir

Package Manager apt

- With access to root, you can install and update software packages
- Ubuntu uses the Debian package manager `dpkg`
- `apt` Makes working with `dpkg` easier
- `apt update` Update package index
- `apt upgrade` Update all installed packages
- `apt install PACKAGE` Install package
- Always run `apt update` before using the other commands
- GWDG HPC-Cluster uses Scientific Linux with `yum` as package manager

Shell Scripting

- Bash commands can be used to program shell scripts
- Written in plain text and saved as `.sh` files
- Must have as first line `#!/usr/bin/bash`
- You can use loops, conditions and so on
  like a regular programming language
- Make it executable if it isn't `chmod +x script.sh`
- Run a script using `./script.sh`
- First inspect a script `less script.sh` or `nano script.sh`
  before running it
- Commonly used to start jobs on supercomputers

Downloading things from the internet

- `wget URL` Download a file at the target URL and save it to disk
- `wget -O NAME URL` Download a file and set its name
- `curl URL` Download a file at the target URL and show it in shell
- `curl -o NAME URL` Download file at URL to a file with name
- Both curl and wget support HTTP(S) and FTP
- curl also supports other protocols and making custom requests
- Common compressions of downloadable files: `tar.gz` or `zip`
- `tar -xzvf FILE.tar.gz` Extracts contents of file to local folder
- `unzip FILE.zip` Extract contents of file to local folder

Git Version Control System (VCS) 1/3

- VCS used for later student projects
- Git repository $\rightarrow$ Folder with files tracked by Git
- Git commit $\rightarrow$ Checkpoint of changes to files
- Manage history of repository in terms of commits
- Share code via remote Git repositories
- Changes can be **Staged** via `git add` to add them to the next commit
- Workflow: Make changes to files $\rightarrow$ Stage changes $\rightarrow$ Commit changes
- Git has many features for collaborative software development

Git Version Control System (VCS) 2/3

- Either use `https://gitlab-ce.gwdg.de` or `https://github.com`
- `git --version` Check version
- `git init` Create a new repository in this folder
- `git status` What files can be added/were added
- `git add TARGET` Add file or directory to **staging**
- `git add .` Add current folder and all sub-folders to **staging**
- `git commit -m "MESSAGE"` Commit all added files with given message
- `git log` See past commits and their messages

Git Version Control System (VCS) 3/3

- `git remote add origin https://gitlab-ce.gwdg.de/USERNAME/PROJECT_NAME`
- `git push --set-upstream origin main` Afterwards only `git push`
- Give your team access by inviting them to the project on Gitlab/Github
- On Gitlab, only maintainer can push to main
- `git clone https://gitlab-ce.gwdg.de/USERNAME/PROJECT_NAME`
- `git pull` Updates local repository, automatically merges and applies commits
- Search online or see `https://books.goalkicker.com/GitBook` for tips
- In case you mess up `https://ohshitgit.com`

Make Command Automation

- `make` Executes commands defined in `makefile`
- `nano makefile` To create a new makefile
- Running `make` executes all commands after `all`
- Must use **tabs** before commands in group, not spaces
- Great for automating builds, compile source, run tests, clean up
- Enter this into a file and try running `make` , `make c3`

```
1 all: c1 c2
2 c1:
3     echo "command 1"
4 c2:
5     echo "command 2"
6 c3:
7     echo "command 3"
```

Postface

- Linux networking was not covered
- Git for Windows comes with the Git Bash shell, which contains most Bash commands `https://gitforwindows.org/`
- Terminal under MacOS uses either Bash or Zsh by default, check your shell with `echo $SHELL` and the version of Bash with `bash --version`
- Find more Bash tricks `https://github.com/dylanaraps/pure-bash-bible`
- Longer guides on Bash, Git and Make:
  - `https://learnxinyminutes.com/docs/bash`
  - `https://learnxinyminutes.com/docs/git`
  - `https://learnxinyminutes.com/docs/make`
- Detailed command lookup `https://explainshell.com/`