

Kõik osade spikrid

See koondspikker koondab peatükkide lõpus olevad täisspikrid osade kaupa ühte faili. Koostamise aeg: **2026-10-02 19:25:09 CEST**.

Osa I: Esimesed sammud — täisspikker

Tase: **Algaja**

Kuidas seda õpikut kasutada

Eesmärk: Saa aru, mis on terminal, shell, CLI ja GUI, ning vali rahulik algusrada enne keerulisemaid töövooge.

Põhitee

- Terminali esimesed sammud — alusta siit
- Abi leidmine — otsi tuge
- Kataloogid ja failid — tee muudatusi
- Teksti vaatamine — loe rahulikult

Tüüpilised kujud

- `pwd` — kontrolli asukohta
- `ls` — vaata sisu
- `man ls` — loe abi
- `cd ~/tmp` — mine ajutiste failide kataloogi; sobib harjutamiseks

Põhimõisted

- CLI — käsurida
- GUI — graafiline liides
- `terminal` — aken käsureaks
- `shell` — tõlgendab käske

Pane tähele: Sa ei pea esimeses ringis veel teadma Giti, Dockerit ega keerulisemaid töövooge.

Õpitee ja õppetunnid

Eesmärk: Vali eeldustele vastav rada, loe peatükke järjest ja kontrolli oskusi harjutusega.

Õpiteed

- Täiesti algaja — loe esimesena
- Süsteemi pildi loomine — pärast esimest ringi
- Igapäevane Linuxi ja serveri kasutaja — kaugühendus ja failid

- Tekst, filtrid ja automatiseerimine — käsurea jõud
- Arendaja suund — Git, Docker, koostamine
- Andmeteaduse stardirada — failid, vormingud, SQL

Kuidas õppida

- Eeldused — mida peaksid enne oskama
- Lugemisjärjekord — liigu loendis ülalt alla
- Harjutus — proovi uut töövõtet eraldi katsekataloogis
- Nüüd oskad — kontrolli, kas saad ülesandega iseseisvalt hakkama

Pane tähele: Ära püüa kogu raamatut korraga läbi võtta; vali kõigepealt üks rada ja liigu selle sees rahulikult edasi.

Terminali esimesed sammud

Eesmärk: Vaata kõigepealt rahulikult, kus sa oled, kes sa oled ja mis selles kataloogis on; alles siis tee esimene väike muudatus.

Käsud ja esimesed sammud

- `pwd` — vaata asukohta
- `ls` — vaata sisu
- `whoami` — vaata kasutajat
- `date` — vaata praegust kuupäeva ja aega
- `history` — vaata käsuajalugu
- `cd ~/tmp` — mine ajutiste failide kataloogi; sobib harjutamiseks
- `mkdir proov` — loo uus kataloog
- `cd proov` — sisene loodud kataloogi

Olulisemad lipud, märgid ja kiirnupud

- `Tab` — lõpeta nimi
- `Ctrl-c` — katkesta programmi töö
- `Ctrl-r` — otsi ajaloost
- `Ctrl-a` — rea algus
- `Ctrl-e` — rea lõpp
- `~/proov %` — näide viibast ehk promptist

Pane tähele: Ära kopeeri viipa käsu ette kaasa; kopeeri ainult käsk ise.

Abi leidmine: `man`, `-help` ja `info`

Eesmärk: Leia tundmatu käsu tähendus ja oma süsteemis toetatud valikud.

Käsud ja näited

- `man ls` — lehitsetav käsiraamat; alusta jaotistest `NAME` ja `SYNOPSIS`

- `ls --help` — GNU `ls`-i kiire abi; macOS-i `ls` kasutab man-lehte
- `whatis ls` — käsu ühe rea kirjeldus
- `apropos archive` — otsi käsiraamatute kirjeldustest
- `info coreutils` — GNU põhikäsud, kui dokumentatsioon on paigaldatud

Märgid ja valikud

- `q` — välju man-i lehitsejast
- `/tekst` — otsi man-lehelt; kinnita Enteriga
- `n / N` — järgmine / eelmine otsinguvastus

Pane tähele: `-help` ega `-h` pole universaalsed; kontrolli konkreetse programmi juhendit.

Kataloogid ja failid

Eesmärk: Failitöö põhiharjumus on: kontrolli asukohta, tee üks väike muudatus, vaata tulemus üle ja kustuta alles pärast teadlikku pausi.

Põhikujud

- `pwd` — kontrolli asukohta
- `ls` — vaata sisu
- `mkdir -p ~/tmp/faili-naited` — loo terve puuduv tee
- `cd ~/tmp/faili-naited` — mine harjutama
- `touch fail.txt` — loo puuduv fail või uuenda ajatempleid
- `cp algne.txt koopia.txt` — tee koopia
- `mv vana.txt uus.txt` — nimeta ümber
- `mv fail.txt kaust/` — liiguta kataloogi
- `rm fail.txt` — kustuta fail
- `rmdir tyhi-kataloog` — eemalda tühi kataloog

Olulisemad lipud, märgid ja kiirnupud

- `.` — praegune kataloog
- `..` — ülemkataloog
- `~` — kodukataloog
- `ls -a` — ka peidetud
- `ls -A` — peidetud ilma `.` ja `..` kirjeteta
- `ls -la` — detailne vaade
- `"fail nimi.txt"` — nimi tühikuga
- `rm -r` — rekursiivne kustutus

Pane tähele: Enne `rm` või `rm -r` tee väike paus: kontrolli `pwd`, vaata `ls`, alles siis kustuta; `rm` ei vii faili prügikasti.

Teksti vaatamine ja liikumine

Eesmärk: Vali faili pikkuse ja küsimuse järgi: kogu sisu, lehitsemine, algus või lõpp.

Käsud ja näited

- `cat /etc/hosts` — kuva lühikese faili sisu
- `less /etc/services` — lehitse pikemat faili
- `head -n 7 fail.txt` — näita esimesed seitse rida
- `tail -n 7 fail.txt` — näita viimased seitse rida
- `tail -f app.log` — oota lisanduvaid ridu; lõpeta Ctrl-c-ga
- `less a.txt b.txt` — ava mitu nimetatud faili

Märgid ja valikud

- `q` — less: välju
- `/tekst` — less: otsi edasi
- `n / N` — less: järgmine / eelmine vaste
- `78g / 25%` — less: rida 78 / veerandi peale
- `g / G` — less: algusse / lõppu
- `v` — less: ava määratud redaktor; pärast väljumist naased
- `:n / :p` — less: järgmine / eelmine fail

Pane tähele: `cat` ei peata pikka väljundit ekraanitäite kaupa.

Failide muutmine: nano ja esimene kokkupuude vim-iga

Eesmärk: less-iga vaatasid faili; tekstiredaktoriga muudad seda. Alguseks sobib nano, mille klahvijuht on akna allservas. `vi` ja `vim` on klassikalised tekstiredaktorid, kus sisestamine ja käskude...

Põhikujud

- `less fail.txt` — vaata faili sisu mugavalt
- `nano fail.txt` — lihtne tekstiredaktor
- `vim fail.txt` — klassikaline redaktor
- `vi fail.txt` — sama pere variant
- `command -v nano` — kontrolli olemasolu

vi/vim minispikker

- `Esc` — välju sisestusest
- `:q` — välju
- `:q!` — välju salvestamata
- `:wq` — salvesta ja välju

Pane tähele: Kui nano puudub või vim hirmutab, siis peamine algaja oskus on see, et sa oskad redaktorist kindlalt väljuda.

Käskude kuju ja argumentide loogika

Eesmärk: Õpi ühe näite pealt eristama käsku, lippu ja argumenti; siis ei tundu uus käsk iga kord täiesti uus keel.

Üks näide

- `ls -la /etc` — terve kuju
- `ls` — käsk
- `-la` — lipud
- `/etc` — argument
- `cp vana.txt uus.txt` — kaks argumenti

Tüüpilised kujud

- `head -n 3 fail.txt` — valik -n vajab arväärtust
- `cp 'fail nimi.txt' koopia.txt` — nimi tühikuga
- `touch -- -algab-kriipsuga.txt` — ära tõlgenda lipuna
- `ls *.txt` — kõik .txt lõpuga failid
- `ls words.0?.txt` — 0 järel üks märk, mitte tingimata number

Olulised märgid

- `-` — lühike lipp
- `--` — üksinda lõpetab valikute loendi
- `"..."` — hoia koos; muutujad toimivad
- `'...'` — täht-tähele nagu kirjas
- `*` — failimuster
- `?` — üks märk

Pane tähele: Kui nimes on tühikuid või erimärke, pane nimi jutumärkidesse; see on üks kõige tavalisemaid algaja komistuskohi.

Sisend, väljund, torud ja suunamine

Eesmärk: Ühenda käske nii, et ühe väljund saab järgmise sisendiks, või saada väljund faili; vea- ja tavaväljundit tasub mõelda eraldi.

Põhikäsud

- `echo` — prindi tekst
- `printf` — vorminda tekst
- `cat` — näita faili
- `tee` — ekraan ja fail
- `wc` — loe kokku

Tüüpilised kujud

- `käsk > valjund.txt` — stdout faili

- `käsk >> logi.txt` — lisa stdout faili lõppu juurde
- `käsk 2> vead.txt` — stderr eraldi
- `käsk > valjund.txt 2> vead.txt` — eralda mõlemad
- `käsk 2> /dev/null` — peida veateated
- `cat fail.txt | wc -l` — toru loendus
- `käsk ; järgmine` — järgmine alati
- `käsk && järgmine` — järgmine ainult edul
- `käsk || varuplaan` — varuplaan vea korral
- `käsk > koik.txt 2>&1` — stdout ja stderr koos

Olulisemad lipud, märgid ja kiirnupud

- `>` — stdout faili
- `>>` — lisa stdout faili lõppu juurde
- `|` — anna edasi järgmisele
- `;` — käivita lihtsalt järjest
- `&&` — tee edasi edu korral
- `||` — tee edasi vea korral
- `2>` — stderr eraldi faili
- `2>&1` — stderr samasse kohta
- `/dev/null` — must auk väljundile

Pane tähele: Õpi kõigepealt rahulikult `>`, `»` ja `|`; alles siis lisa sellele `&&`, `||` ja `2>&1`.

Esimene tervikharjutus: 30 minutit

Eesmärk: kontrolli alguskohta; tööta eraldi harjutuskataloogis; lisa faili read väikeste sammudena

Lühike töövoog

- `pwd && ls` — kontrolli alguskohta
- `mkdir proov && cd proov` — loo töökataloog
- `echo tere > sonad.txt` — tee esimene rida
- `echo maailm >> sonad.txt` — lisa järgmine rida
- `cat sonad.txt && wc -l sonad.txt` — kontrolli tulemust
- `cp sonad.txt koopia.txt` — tee koopia

Olulised märgid

- `>` — kirjuta üle
- `>>` — lisa faili lõppu juurde
- `Ctrl-c` — katkesta programmi töö
- `man` — vaata abi

Pane tähele: Kui mõni käsk on vahepeal hägusaks läinud, peatu ja vaata `man` või eelmised Osa I peatükid üle.

Osa II: Süsteemi pilt ja haldus — täisspikker

Tase: Baas ja süsteemipilt

Linux, Unix, GNU, macOS, Windows ja shellid

Eesmärk: erista süsteemi, shelli ja käsureatööriistu; sama terminaliaken ei tähenda veel, et Linux, macOS ja Windows käituksid ühtemoodi

Kontrollkäsud

- `uname -a` — süsteemi ja kerneli rida
- `echo "$SHELL"` — seadistatud sisselogimisshell
- `ps -p $$` — praegu töötav shelliprotsess
- `command -v ls` — kust käsk leitakse
- `sw_vers` — macOS-i versioon
- `wsl -l -v` — WSL-i jaotused

Olulised mõisted

- Linux — kernel ehk tuum
- GNU — paljud põhikäsud
- shell — tõlgendab kāske
- WSL — Linux Windowsis
- BSD — macOS-i tööriistapere
- PowerShell — Windowsi käsukeskkond

Pane tähele: macOS-is ja Windowsis võivad sama nimega käsud käituda teistmoodi kui Linuxis.

Failisüsteemi kaart

Eesmärk: õpi eristama süsteemi juurt, oma kodukataloogi ning ajutisi ja süsteemseid katalooge; siis on lihtsam aru saada, kus tohid rahulikult katsetada

Põhikujud

- `pwd` — kontrolli asukohta
- `echo "$HOME"` — vaata oma kodu
- `cd /` — mine juurkataloogi
- `cd ~` — mine kodukataloogi
- `ls /etc | head` — piilu seadistusi
- `ls -ld /tmp` — vaata ajutist kataloogi

Olulised teed

- `/` — kogu failipuu juur
- `~` — sinu kodukataloog

- `/etc` — süsteemi seadistused
- `/var` — logid ja muutuv sisu
- `/tmp` — ajutiste failide koht
- `/Users` / `/home` — kasutajate kodud

Pane tähele: `/` ja `~` ei ole sama asi: esimene on süsteemi juur, teine sinu enda tööala.

Kettaruum ja süsteemi maht

Eesmärk: `df` näitab failisüsteemi vaba ruumi, `du` loendab valitud failide ja kataloogide kettakasutust.

Käsud ja näited

- `df -h .` — selle failisüsteemi kasutatud ja vaba maht
- `du -sh .` — praeguse kataloogipuu kogumaht
- `du -h -d 1 .` — esimese taseme mahud, ka peidetud kataloogid
- `du -h -d 1 . | sort -h` — sordi ühikusufiksiga mahud kasvavalt
- `du -ak . | sort -nr | less` — kõik kirjed 1024-baidistes ühikutes, suurimad ees

Märgid ja valikud

- `du -s` — ainult kogusumma
- `du -a` — ka üksikfailide mahud
- `du -k` — ühtlane 1024-baidine ühik
- `sort -h` — arvestab ühikusufiksi järku, siis arvvaartust
- `sort -nr` — arvuliselt kahanev järjestus

Pane tähele: Veateated võivad tähendada puudulikku kogusummat. Ära liida `du` alamkataloogide ja nende vanemate mahtusid uuesti kokku.

Õigused, omanikud ja täitmisbitid

Eesmärk: loe õiguste rida, mõista `rwx` tähendust ja erista faili käivitatavust kataloogi läbikäigust

Põhikujud

- `ls -l fail.txt` — loe õiguste rida
- `chmod u+x skript.sh` — anna omanikule täitmine
- `chmod 644 naide.txt` — sea tekstifaili õigused
- `chmod 755 skript.sh` — sea käivitatava õigused
- `sudo chown kasutaja:grupp fail.txt` — muuda omanikku

Olulised õigusekujud

- `r` — loe sisu
- `w` — muuda sisu
- `x` — käivita või sisene
- `d` — tegu on kataloogiga
- `644` — omanik `rw`, teised `r`
- `755` — omanik `rw`, teised `rx`

Pane tähele: Kataloogi `x`-õigus tähendab läbikäiku; see ei ole sama asi mis faili käivitamine.

Kasutajad, grupid ja sudo

Eesmärk: õpi nägema, millise kasutajana sa töötad, millistes gruppides oled ja millal on sudo päriselt vajalik

Põhikujud

- `whoami` — kontrolli kasutajat
- `id` — loe grupid välja
- `groups` — vaata grupinimesid
- `sudo -l` — vaata sudo õigusi
- `sudo apt update` — üks kõrgem käsk
- `su - kasutaja` — vaheta kasutajat

Olulised märgid ja rollid

- `$` — tavaline kasutaja
- `%` — teine levinud tavaviip
- `#` — root või kõrgemad õigused
- `root` — administraatori kasutaja
- `sudo -i` — ava kõrgem shell
- `sudo käsk` — eelista ühte käsku

Pane tähele: Ära kasuta `sudo`-t lihtsalt harjumusest; enne kontrolli, kas käsk tõesti vajab kõrgemaid õigusi.

Muutujad, keskkond, PATH ja aliased

Eesmärk: erista shellimuutujat, keskkonnamuutujat, käsuotsingu teed ja aliaist; `PATH`-i järjekord otsustab, milline väline käsk esimesena leitakse.

Põhikujud

- `nimi='Mari'; echo "$nimi"` — shellimuutuja kehtib selles shellis
- `export DEMO=1` — anna muutuja edasi käivitatavatele programmidele
- `env | grep DEMO` — kontrolli, kas muutuja on keskkonnas
- `echo "$PATH"` — vaata käskude otsinguteed

- `command -v python3` — näita käsu lahendust, ka alias või sisekäsk
- `type cd` — vaata käsu liiki
- `type -a grep` — näita kõik vasted
- `export PATH="$HOME/bin:$PATH"` — pane oma käskude kataloog otsingutee ette
- `alias ll='ls -lah'` — loo lühinimi sagedasele käsule
- `source ~/.zshrc` — loe zsh seadistus uuesti

Olulised mõisted

- `nimi='Mari'` — shellimuutuja
- `export NIMI='Mari'` — keskkonnamuutuja
- `PATH` — otsinguteede järjekord
- `$HOME/bin` — oma käskude koht
- `type` — alias, sisekäsk või programm
- `~/.zshrc` — zsh seadistusfail
- `~/.bashrc` — bash seadistusfail

Pane tähele: `PATH`-i järjekord loeb: vasakul olev vaste leitakse enne; ära lisa tundmatut kataloogi ettepoole lihtsalt sellepärast, et juhend nii ütleb.

Paketihaldus: `apt`, `dnf`, `pacman`, `brew`

Eesmärk: vali õige tase: süsteemi paketihaldur tööriistadele, keele paketihaldur teekidele ja projekti sõltuvused võimalusel projekti sisse

Põhikujud

- `apt search ripgrep` — otsi Debianis või Ubuntu
- `sudo apt install ripgrep` — paigalda Debianis või Ubuntu
- `sudo dnf install ripgrep` — paigalda Fedoras
- `sudo pacman -S ripgrep` — paigalda Archis
- `brew install ripgrep` — paigalda Homebrew'ga
- `python3 -m pip install requests` — paigalda Pythoni pakett aktiivsesse keskkonda
- `npm install` — paigalda projekti Node.js sõltuvused
- `command -v rg` — kontrolli, kas käsk leitakse

Tööriistad ja tasemed

- `apt`, `dnf`, `pacman` — Linux'i süsteemipaketid
- `brew` — macOS-i kasutajataseme tööriistad
- `update` — värskenda nimekirja või retsepte
- `upgrade` — uuenda paigaldatud paketid
- `--cask` — Homebrew graafiline rakendus
- `venv` — Pythoni projekti eraldi keskkond
- `package.json` — Node.js projekti sõltuvused

Pane tähele: Ära aja segi süsteemi paketihooldurit ja pip/npm-i; need lahendavad eri taseme probleeme.

Lihtne veaotsing käsuraal

Eesmärk: kitsenda viga enne parandamist: täpne veateade, asukoht, olemasolu, käsu leitavus, õigused ja shelli/süntaksi kontroll

Kontrollrada

- `pwd` — kontrolli, kus oled
- `ls -lah` — vaata faile, õigusi ja peidetud kirjeid
- `command -v rg` — kas käsk leitakse
- `type -a rg` — näita kõik sama nimega vasted
- `ls -l skript.sh` — vaata skripti õigusi
- `head -n 1 skript.sh` — vaata skripti shebang-rida
- `bash -n skript.sh` — kontrolli süntaksit ilma käivitamata
- `grep --help` — kinnita käsu süntaks

Levinud veateated

- `command not found` — käsk puudub, nimi vale või pole PATH-is
- `No such file or directory` — tee, failinimi või asukoht on vale
- `Permission denied` — õigus ei luba lugeda, kirjutada või käivitada
- `syntax error` — vale shell või vale käsukuju
- `package not found` — paketinimi või nimekiri on vale
- `database is locked` — andmebaasi hoiab teine toiming; põhjus sõltub programmist

Pane tähele: Pane kõigepealt kirja täpne käsk ja täpne veateade; ilma nendeta muutub veaotsing oletamiseks.

Võrgu põhitooriistad

Eesmärk: erista nelja küsimust: kas host vastab, kas veeb vastab, millised liidesed on ja kas kohalik teenus kuulab porti

Kontrollrada

- `curl -I https://example.com/` — kontrolli veebiteenust
- `ping -c 4 example.com` — kontrolli hosti ICMP vastust
- `ip a` — vaata Linux liideseid
- `ifconfig` — vaata macOS-i liideseid
- `ss -ltn` — vaata Linux kuulavaid TCP-porte
- `lsof -iTCP -sTCP:LISTEN -n -P` — vaata macOS-i kuulavaid porte
- `nc -vz example.com 443` — kontrolli TCP ühendust porti

Olulised võtmed ja vaated

- `curl -I` — küsi HTTP-päised HEAD-päringuga
- `curl -L` — järgi HTTP ümbersuunamisi
- `LISTEN` — port kuulab
- `127.0.0.1` — ainult oma masin
- `0.0.0.0` — kõik IPv4-liidesed
- `ip/ss` — Linuxi tööriistad
- `ifconfig/lsof` — macOS-i põhivariandid

Pane tähele: ping ja HTTP ei ole sama asi; veeb võib töötada ka siis, kui host pingile ei vasta.

Osa III: Failid, võrk ja süsteemitöö — täisspikker

Tase: **Praktiline kasutus**

Failide kopeerimine ja sünkroonimine

Eesmärk: vali tööriist allika järgi: `cp` samas masinas, `scp` üle SSH, `rsync` korduvaks sünkroonimiseks ja `curl/wget` veebist toomiseks

Põhikujud

- `cp fail.txt koopia.txt` — tee koopia
- `cp -R kaust kaust-koopia` — kopeeri kataloog
- `scp fail.txt kasutaja@server:/tmp/` — saada serverisse
- `rsync -avn kaust/ kasutaja@server:/tmp/kaust/` — kontrolli enne
- `rsync -av kaust/ kasutaja@server:/tmp/kaust/` — sünkrooni sisu
- `wget https://artificial.eu/linux/data/generated-words.txt` — laadi fail
- `curl -fLO https://artificial.eu/linux/data/generated-words.txt` — järgi suunamisi, HTTP-vea korral ebaõnnestu

Olulisemad lipud, märgid ja kiirnupud

- `cp -R` — kopeeri kataloog
- `scp -r` — saada kataloog
- `rsync -a` — rekursiivselt, säilitades tavalised Unixi metaandmed
- `rsync -n` — proovi enne
- `curl -L` — järgi suunamist
- `curl -O` — salvesta URL-i lõpuosa nimega

Pane tähele: Vali tööriist allika järgi: `cp` samas masinas, `scp` üle SSH, `rsync` korduvaks sünkroonimiseks, `curl/wget` veebist toomiseks.

Kauglogimine ja SSH

Eesmärk: tee SSH korduvkasutatavaks: kontrolli olemasolevaid võtmeid, lisa avalik võti serverisse, kasuta failis ~/.ssh/config määratud aliaast ja macOS-is Apple'i võtmehoidjat

Põhirada

- `ssh kasutaja@server.example.org` — proovi tavalist sisselogimist
- `ls -la ~/.ssh` — vaata olemasolevaid võtmefaile
- `ssh-add -l` — vaata agendis olevaid võtmeid
- `ssh-keygen -t ed25519 -C 'kasutaja@arvuti'` — loo võti ainult vajadusel
- `/usr/bin/ssh-add --apple-use-keychain ~/.ssh/id_ed25519` — macOS: salvesta võtmefraas võtmehoidjasse
- `ssh-copy-id -i ~/.ssh/id_ed25519.pub kasutaja@server.example.org` — lisa avalik võti serverisse
- `ssh opik-server` — kasuta seadistusfaili aliaast
- `ssh -v opik-server` — diagnostika

Olulised failid ja mõisted

- `id_ed25519` — privaatvõti; ära jaga
- `id_ed25519.pub` — avalik võti serverisse
- `~/.ssh/config` — ühenduse aliaast
- `~/.ssh/authorized_keys` — lubatud võtmed
- `known_hosts` — serverite sõrmejäljed
- `AddKeysToAgent` — lisa võtmed agenti
- `UseKeychain` — võtmefraas macOS-i võtmehoidjas
- `IdentitiesOnly yes` — kasuta nimetatud võtit

Pane tähele: Alusta sisselogimisest ja võtmetest; port forwarding nagu `ssh -L` ... on järgmine, mitte esimene samm.

Veebist sisu toomine ja tekstivaade: curl, wget, lynx

Eesmärk: vali curl ühe vastuse või päiste jaoks, wget allalaadimise jaoks ja lynx HTML-i rahulikuks tekstivaateks

Põhikujud

- `curl -I https://example.com/` — kontrolli URL-i
- `curl -L -o leht.html https://example.com/` — salvesta HTML
- `wget -O leht.html https://example.com/` — laadi faili
- `curl -fsSL https://example.com/ | lynx -stdin -dump` — loe tekstina
- `lynx -dump -listonly https://example.com/` — kogu lingid

Olulisemad lipud, märgid ja kiirnupud

- `curl -I` — vaata päiseid
- `curl -L` — järgi suunamist
- `curl -O` — salvesta URL-i lõpuosa nimega
- `wget -c` — jätkata tõmmet
- `lynx -dump` — tekstvaade
- `lynx -listonly` — ainult lingid

Pane tähele: Alusta alati ühest URL-ist; ulatuslik veebis roomamine ja automaatne veebikraapimine olgu järgmine, mitte esimene samm.

Arhiivid ja pakkimine

Eesmärk: arhiiv koondab failid; tihendamine võib mahtu vähendada, kuid juba pakitud sisu puhul mitte tingimata

Põhikujud

- `tar -cf proov.tar kaust/` — loo arhiiv
- `tar -tf proov.tar` — vaata sisu
- `tar -czf proov.tar.gz kaust/` — loo gzip-arhiiv
- `tar -xzf proov.tar.gz -C siht/` — paki lahti sihtkohta
- `zip -r proov.zip kaust/` — loo zip
- `unzip -l proov.zip` — vaata zip sisu

Olulisemad lipud, märgid ja kiirnupud

- `tar -cf` — loo arhiiv
- `tar -tf` — vaata sisu
- `tar -czf` — gzip-iga kokku
- `tar -xzf` — paki gzip lahti
- `zip -r` — paki kataloog ZIP-arhiivi
- `unzip -l` — vaata zip sisu

Pane tähele: Enne lahtipakkimist vaata võimalusel arhiivi sisu; see aitab näha, kas sees on oodatud juurkataloog ja failipuud.

Tervete kataloogipuude haldus ja jagamine

Eesmärk: terve projekti või kataloogipuu puhul vali tööriist selle järgi, kas tahad koopiat, korduvat sünkroonimist, arhiivi või versioonijalugu

Põhikujud

- `cp -R projekt projekt-koopia` — tee kohalik koopia
- `rsync -avn projekt/ projekt-varu/` — kontrolli enne
- `rsync -av projekt/ kasutaja@server:/srv/projekt/` — sünkrooni serverisse

- `scp -r projekt kasutaja@server:/tmp/` — saada kiirelt
- `tar -czf projekt.tgz projekt/` — tee hetkearhiiv
- `git` — jälgi ajalugu

Olulisemad lipud, märgid ja kiirnupud

- `projekt/` — rsync-allikas: kataloogi sisu
- `projekt` — rsync-allikas: kataloog ise
- `rsync -n` — proovi enne
- `cp -a` — säilita metaandmed

Pane tähele: Kui vajad korduvat sünkroonimist, eelista tavaliselt rsync-i; scp -r sobib pigem üheks kiireks ülekandeks.

Protsessid, tööd ja signaalid

Eesmärk: erista protsessi, shelli tööd ja signaali: vaata, millised protsessid töötavad, peata vajadusel ning jätkka pikka tööd taustal

Põhikujud

- `ps aux | grep python` — otsi protsess
- `ps aux | sort -nrk 3 | head` — suur CPU kasutus selles hetkepildis
- `ps aux | sort -nrk 4 | head` — suur mälu kasutus
- `sleep 300 &` — käivita taustal
- `jobs` — vaata töid
- `fg %1` — too töö ette
- `top` — jälgi reaalajas
- `kill` — saada signaal
- `bg` — jätkka taustal

Olulisemad lipud, märgid ja kiirnupud

- `Ctrl-c` — katkesta programmi töö
- `Ctrl-z` — peata ajutiselt; fg ette, bg taustale
- `&` — käivita taustal
- `%1` — töö number
- `kill -15` — viisakas lõpp
- `kill -9` — jõuga lõpp

Pane tähele: Kasuta `kill -9` alles siis, kui tavalisem `kill` või `kill -15` ei lõpeta protsessi viisakalt.

Logid ja teenused

Eesmärk: Kontrolli teenuse olekut ja seejärel sama teenuse logi. `systemctl/journalctl` vajavad `systemd`-ga Linuxit.

Käsud ja näited

- `systemctl status ssh` — teenuse seis ja hiljutised teated; nimi võib olla `sshd`
- `journalctl -u ssh -n 50` — selle teenuse viimased 50 kirjet
- `journalctl -u ssh --since today` — tänased teenusekirjed
- `journalctl -u ssh -f` — jälgi juurde tulevaid kirjeid
- `tail -n 20 app.log` — tavalise logifaili lõpp, ka ilma `systemd`-ta
- `tail -f app.log` — jälgi faili; `Ctrl-c` lõpetab jälgimise

Pane tähele: Logi puudumine ei tõesta vigade puudumist: kontrolli teenuse nime, logimise seadistust ja lugemisõigust.

Püsivad terminalisessioonid: `tmux` ja `screen`

Eesmärk: Säilita interaktiivne serverisessioon ka SSH-ühenduse katkemisel.

Käsud ja näited

- `tmux new -s katse` — loo nimega sessioon
- `tmux ls` — loetle sessioonid välisest shellist
- `tmux attach -t katse` — ühendu olemasoleva sessiooniga
- `screen -S katse` — alternatiiv: loo screeni sessioon
- `screen -r katse` — taasühendu eraldatud screeniga

Märgid ja valikud

- `Ctrl-b`, siis `d` — `tmux`: eraldu, töö jätkub
- `Ctrl-a`, siis `d` — `screen`: eraldu, töö jätkub
- `exit` — lõpeta praeguse akna shell

Pane tähele: Sessioon ei ela üle serveri taaskäivitust ega volukatkestust.

Graafilised rakendused kaugmasinast

Eesmärk: kaugelt graafika kasutamisel eelista lihtsaimat toimivat teed: veebiliides enne X11-edastust, port forwarding enne toorest kaugtöölauda.

Põhikujud

- `ssh -N -L 127.0.0.1:8888:localhost:8888 kasutaja@server` — tunnel juba töötava serveriteenuseni
- `http://localhost:8888` — ava edasi suunatud teenus
- `ssh -X kasutaja@server` — proovi lihtsat X11
- `code --remote ssh-remote+server /tee/projektini` — IDE üle SSH

Valiku rusikareeglid

- veebiliides — tavaliselt mugavam

- `ssh -L` — port edasi
- `ssh -X` — X11 erijuht
- Remote SSH — arenduseks parem

Pane tähele: Kui sul on valida, eelista brauserit või Remote SSH-d; X11 forwarding olgu pigem varuplaan, mitte esimene mõte.

Osa IV: Tekst, otsing ja automatiseerimine — täisspikker

Tase: Töövood

Teksti otsimine: `grep` ja sugulased

Eesmärk: Vali tekstist sobivad read; erista täpset teksti regulaaravaldisest.

Otsingukuju ja tulemus

- `grep 'kass' loomad.txt` — ka pikema sõna sees; väljastab terve rea
- `grep -in 'kass' loomad.txt` — eira tähesuurust, lisa reanumbrid
- `grep -x 'kass' loomad.txt` — ainult tervenisti sobiv rida
- `grep -v '^#' seadistus.conf` — jäta välja `#` märgiga algavad read
- `grep -F '.json' nimed.txt` — punkt tähendab päris punkti, mitte suvalist märki
- `grep -rn 'TODO' .` — otsi failipuust koos failinimede ja reanumbritega
- `grep -c 'kass' loomad.txt` — loenda vastega ridu, mitte kõiki esinemisi

`grep-i` tulemuskoodid

- 0 — vähemalt üks vaste
- 1 — vasteid pole, mitte tingimata viga
- 2 — tavaliselt viga; loe veateadet

Pane tähele: `-F` otsib täpset teksti ka pikema rea seest. Terve rea sobitamiseks lisa `-x`.

Teksti teisendamine: `tr`, `cut`, `paste`, `column`, `strings`

Eesmärk: need väikesed filtrid muudavad tekstivoo kuju kiiresti: märgid, väljad, veerud ja tabelid ilma eraldi skriptita

Põhikujud

- `echo 'tere maailm' | tr '[:lower:]' '[:upper:]'` — muuda suurtäheks
- `echo 'a,b,c' | tr ',' '\n'` — tee ridadeks

- `cut -d ':' -f 1 andmed.txt` — võta esimene väli
- `cut -d ':' -f 1,3 andmed.txt` — võta mitu välja
- `printf 'nimi vanus\nMari 20\n' | column -t` — joonda tabel
- `strings /bin/ls | head -n 20` — loe binaarist tekst
- `paste` — kleebi veerge

Olulisemad lipud, märgid ja kiirnupud

- `[:lower:]` — väiketähed
- `[:upper:]` — suurtähed
- `cut -d` — vali eraldaja
- `cut -f` — vali väljad
- `column -t` — joonda tabel
- `strings` — tekstijupid

Pane tähele: Kui väljund muutub imelikuks, kontrolli kõigepealt eraldajat ja sisendi kuju; need tööriistad on väikesed, aga väga sõna-sõnalt loetavad.

Vood ja tabelid: `sort`, `uniq`, `wc`, `pr`, `join`

Eesmärk: tüüpiline töövoog on: sorteerid read, koonda kordused ja loe tulemused kokku

Põhikujud

- `sort viljad.txt` — sordi read
- `sort viljad.txt | uniq` — eemalda kordused
- `sort viljad.txt | uniq -c` — loe kordused
- `wc -l data/sample-text.txt` — loe read
- `wc -w data/sample-text.txt` — loe sõnad
- `join nimed.txt linnad.txt` — ühenda võtme järgi
- `pr` — jaga veergudeks

Olulisemad lipud, märgid ja kiirnupud

- `sort -n` — numbrid
- `sort -r` — tagurpidi
- `uniq -c` — loenda kordused
- `wc -l` — ridade arv
- `wc -w` — sõnade arv
- `pr -2` — kaks veergu

Pane tähele: `uniq` ühendab ainult kõrvutised võrdsed read. Kui algjärjekord pole oluline, koonda võrdsed read esmalt käsuga `sort`.

sed, awk ja perl praktiliselt

Eesmärk: kasuta sed lihtsaks asenduseks, awk väljade jaoks ja perl siis, kui vajad tugevamat regulaaravaldiste loogikat

Põhikujud

- `echo 'kass koer kass' | sed 's/kass/rebane/'` — asenda esimene
- `echo 'kass koer kass' | sed 's/kass/rebane/g'` — asenda kõik
- `printf 'Mari:20\nJaan:21\n' | awk -F: '{print $1}'` — võta esimene väli
- `printf 'Mari:20\nJaan:21\n' | awk -F: '{print $1, $2}'` — võta kaks välja
- `echo 'kass koer kass' | perl -pe 's/kass/rebane/g'` — asenda Perliga
- `printf 'Mari\nJaan\n' | perl -ne 'print if /Ja/'` — filtreeri Perliga

Olulisemad lipud, märgid ja kiirnupud

- `s/vana/uus/` — üks asendus
- `/g` — kõik vasted
- `-F:` — koolon väljade vahel
- `$1` — esimene väli
- `$2` — teine väli
- `-pe / -ne` — read läbi

Pane tähele: Kui vajad ainult üht lihtsat asendust, alusta sed-ist; ära hüpka kohe awk või perl juurde enne, kui lihtsam tee on ammendunud.

find ja xargs ohutumalt

Eesmärk: otsi failid find-iga ja anna need ohutult edasi järgmisele käsule, isegi siis, kui nimedes on tühikuid

Põhikujud

- `find . -type f -name '*.txt'` — otsi failid
- `find . -type f -name '*.txt' -print0 | xargs -0 wc -l` — loe read ohutult
- `find . -type f -name '*.log' -exec ls -lh {} +` — käivita otse
- `find data -type f -name '*.txt' -print0 | while IFS= read -r -d '' fail; do head -n 1 "$fail"; done` — töötle ükshaaval

Olulisemad lipud, märgid ja kiirnupud

- `-type f` — find: ainult tavafailid
- `-name '*.txt'` — nime järgi

- `-print0` — `find`: failinimede vahel nullmärk
- `xargs -0` — `xargs`: eralda argumendid nullmärgiga
- `-exec käsk {} +` — käivita otse
- `read -r -d ''` — Bash/zsh: loe nullmärgini, säilita kaldkriipsud

Pane tähele: Kui failinimes võib olla tühikuid või reavahetusi, eelista paari `-print0` ja `xargs -0`.

Esimene shelliskript

Eesmärk: esimese shelliskripti tuum on viis asja: shebang, käivitatavus, argumentid, lihtne tingimus ja lõpetuskood

Põhikujud

- `chmod +x tervita.sh` — tee käivitatavaks
- `./tervita.sh Mari Jaan` — käivita argumentidega
- `./tervita.sh` — näe veajuhtumit
- `echo $?` — vaata lõpetuskoodi
- `bash skript.sh` — käivita Bashiga
- `sh skript.sh` — käivita sh-ga
- `./skript.sh` — kasuta shebang-rida

Olulisemad lipud, märgid ja kiirnupud

- `#!/usr/bin/env bash` — vali Bash
- `$1` — esimene argument
- `"$@"` — kõik argumendid
- `if ... fi` — tingimus
- `for ... done` — kordus
- `exit 0` / `exit 1` — edu või viga

Pane tähele: Kõige tähtsam vahe on see: `./skript.sh` kasutab shebang-rida, `bash skript.sh` valib tõlgendaja käsurealt.

cron ja ajastatud tööd

Eesmärk: Ajasta läbiproovitud käsk; kasuta täisteid ja talleta väljund logisse.

Käsud ja näited

- `crontab -l` — vaata praeguse kasutaja ajastusi
- `crontab -e` — muuda tabelit, säilitades muud tööd
- `/bin/date >> "$HOME"/cron-naide.log 2>&1` — testi ajastatavat käsku käsitsi
- `tail -n 5 "$HOME"/cron-naide.log` — kontrolli, kas töö lisab kirjeid

Märgid ja valikud

- `*/15 * * * *` — minutitel 0, 15, 30 ja 45
- `0 8 * * 1-5` — tööpäeviti kell 08.00 süsteemi ajavööndis
- `crontab fail` — asendab kogu kasutaja ajastustabeli
- `crontab -r` — eemaldab kogu kasutaja ajastustabeli

Pane tähele: Ära kasuta `crontab` fail ühe rea lisamiseks. Cron ei käivita tavaliselt arvuti uneajal vahele jäänud töid tagantjärele.

Osa V: Arendus ja töövood — täisspikker

Tase: **Edasijõudnu**

Git, GitHub ja töövoog

Eesmärk: Git hoiab muutuste ajalugu; erista tööpuud, vaheala ja versioonikirjet ning vaata muudatused enne üle.

Põhikujud

- `git status` — vaata seis
- `git diff` — muutused tööpuus
- `git diff --cached` — muutused vahealal
- `git add fail.txt` — lisa vahealale
- `git commit -m '...'` — loo versioonikirje
- `git switch -c parandus` — loo haru
- `git pull --ff-only` — uuenda puhtalt
- `git push -u origin parandus` — saada haru
- `git restore --staged fail.txt` — eemalda vahealalt

Olulisemad lipud, märgid ja kiirnupud

- `HEAD` — praegune tipp
- `main` — põhirida
- `origin` — kaugrepo
- `vaheala` — järgmise versioonikirje sisu

Pane tähele: Uus jälgimata fail ei ilmu tavalisse `git diff` vaatesse; pärast `git add` kontrolli esimese versioonikirje sisu käsuga `git diff --cached`.

Pythoni venv ja eraldatud keskkonnad

Eesmärk: `venv` hoiab ühe projekti Pythoni paketid eraldi; kõige olulisem on, et terminal, `pip` ja IDE näeksid sama keskkonda.

Põhikujud

- `python3 -m venv .venv` — loo keskkond
- `source .venv/bin/activate` — aktiveeri shellis
- `python -m pip install -U pip` — uuenda pip
- `python -m pip install requests` — paigalda pakett
- `python -m pip list` — vaata pakke
- `command -v python` — kontrolli tõlgendit
- `deactivate` — välju keskkonnast

Olulised märgid

- `(.venv)` — aktiivne keskkond
- `.venv/` — projekti sees
- `python -m pip` — kindlam kui pip
- Docker — terve keskkond

Pane tähele: Kui pip install läks valesse kohta, kontrolli kohe `command -v python` ja `command -v pip`; enamasti on probleem just vales aktiivses keskkonnas.

Docker'i alused

Eesmärk: Konteinerkuju (image) on käivitusmall; konteiner võib töötada või olla peatatud. Konteineri eemaldamisel kaob selle kirjutatav kiht, mitte seotud hostifailid ega püsivad andmeköited.

Põhikäsud

- `docker --version` — kontrolli olemasolu
- `docker run --rm alpine echo tere` — ohutu esimene käivitus
- `docker build -t rakendus .` — koosta konteinerkuju
- `docker run --rm rakendus` — käivita konteiner
- `docker ps` — töötavad konteinerid
- `docker logs -f nimi` — jälgi logi
- `docker exec -it nimi sh` — sisene töötavasse konteinerisse
- `docker compose up -d` — teenused taustale
- `docker compose down` — peata ja eemalda konteinerid ning võrk

Olulisemad lipud, märgid ja kiirnupud

- `image` — konteinerkuju ehk käivitusmall
- `konteiner` — käivitatud või peatatud eksemplar
- `Dockerfile` — konteinerkuju retsept
- `compose.yaml` — mitu teenust
- `volume` — püsiv andmeköide
- `registry` — konteinerkujude hoidla
- `--rm` — kustuta lühikatske konteiner pärast lõppu

- `-p 127.0.0.1:8000:8000` — docker run: avalda port ainult oma arvutile
- `--mount type=bind,src="$PWD",dst=/app` — jaga praegune kataloog konteinerisse
- `-w /app` — määra konteineri töökataloog
- `docker compose down -v` — peatab ja kustutab ka andmeköited

Pane tähele: Kui konteiner ei tööta, vaata enne docker logs; ära seo tundmatu konteinerkujuga oma kodukataloogi ega salajasi faile.

IDE-d ja arenduskeskkonnad

Eesmärk: heas arenduskeskkonnas osutavad terminal, Git, tõlgendaja ja vajaduse korral konteiner samale projektile.

Kontrollid ja valikud

- `python3 --version` — kontrolli Pythonit
- `python3 -m pip --version` — kontrolli pip-i
- `git --version` — kontrolli Giti
- `docker --version` — kontrolli Dockerit
- `node --version` — kontrolli Node'i
- `npm --version` — kontrolli npm-i

Olulised mõisted

- tõlgendaja — koodi tegelik käivitaja
- sisseehitatud terminal — sama projektivaade
- Remote SSH — tööta kaugmasinas
- devcontainer — IDE konteineris

Pane tähele: Kui IDE-s miski ei tööta, kontrolli kõigepealt, kas IDE kasutab sama Pythonit, Node'i või konteinerit kui terminal.

Andmeteaduse eelteadmised käsurea vaates

Eesmärk: andmeteaduse stardis ei piisa ühest tööriistast; vaja on korraga failivormingute tunnetust, SQL-i mõtteviisi, programmeerimist ja statistilist mõtlemist.

Põhikujud

- `fail -> head/less -> column/jq -> sqlite3 -> python3` — alusta väikese vaatusega
- Pythoni `venv` — projekti töölaud
- CSV, JSON ja XML — andmete kuju
- SQLite ja Python — esimene andmebaas
- Teksti teisendamine — väiksed filtrid
- Vood ja tabelid — koondamine

Põhiteljed

- programmeerimine — korduv töö loogikaks
- SQL — küsi ja seo andmeid
- vormingud — CSV, JSON, XML
- statistika — anna tähendus

Pane tähele: Ära mine otse mudeli või märkmiku (notebook) juurde enne, kui oled vähemalt korra kontrollinud, mis kujul andmed sul tegelikult on.

CSV, JSON ja XML käsureal

Eesmärk: vormingu tüüp ütleb, milline tööriist mõistab andmeid kõige loomulikumalt: tabelit, puud või märgendistruktuuri.

Põhikujud

- `head fail.csv` — vaata CSV algust
- `column -s, -t < fail.csv` — joonda lihtne CSV
- `cut -d, -f1 fail.csv` — võta üks veerg
- `python3 -m json.tool fail.json` — vorminda JSON
- `jq '.students[].name' fail.json` — vali JSON välju
- `xmllint --format fail.xml` — vorminda XML

Olulised mõisted

- CSV — lihtne tabel
- JSON — objektide puu
- XML — märgendipuu
- jq — JSON filtriiks

Pane tähele: Kui CSV-s on jutumärgid, komad välja sees või reavahetused, ei pruugi `cut -d`, enam piisata; siis mine pigem Pythonisse või spetsiaalsema parseri juurde.

Andmebaasi algus: sqlite ja Python

Eesmärk: SQLite on sild tekstifailide ja päris andmebaasimõtte vahel: failina lihtne, SQL-i mõttes siiski relatsiooniline andmebaas.

Põhikujud

- `sqlite3 andmed.db` — ava või loo fail
- `sqlite3 andmed.db '.tables'` — vaata tabeleid
- `sqlite3 andmed.db '.schema'` — vaata struktuuri
- `sqlite3 andmed.db 'select * from students limit 5;'` — piilu ridu
- `sqlite3 andmed.db 'select city, count(*) from students group by city;'` — koonda read

- `sqlite3 andmed.db 'select s.name, r.score from results r join students s on s.id = r.student_id;'` — ühenda tabelid
- `python3 naide.py` — kasuta Pythonist

Olulised mõisted

- `primary key` — rea unikaalne id
- `foreign key` — viide teise tabelisse
- `JOIN` — seo tabelid
- `GROUP BY` — koonda read

Pane tähele: Ära tee `JOIN`-i ainult käsu pärast; enne joonista enda jaoks välja, milline väli viitab millisele tabelile.

Kompileerimine ja käivitamine: `shell`, `Python`, `C`, `C++`, `Go`, `Rust`, `Java`

Eesmärk: Erista lähtekoodi, koostatud väljundit ja selle käivitajat. Vali üks paigaldatud tööriistaga keel.

Käsud ja näited

- `sh hello.sh` — tõlgenda POSIX-shelliskript
- `python3 hello.py` — käivita Pythoni lähtefail
- `cc hello.c -o hello-c` — kompileeri `C` lähtefail käivitatavaks failiks
- `./hello-c` — käivita äsja koostatud `C` programm
- `c++ hello.cpp -o hello-cpp` — kompileeri `C++`
- `go build -o hello-go hello.go` — koosta `Go` binaarfail
- `cargo run` — `Rust`: koosta ja käivita `Cargo` projekt
- `javac Hello.java` — `Java`: loo baitkoodifail `Hello.class`
- `java Hello` — käivita klass `JVM`-is, ilma `.class`-laiendita

Pane tähele: `./fail` vajab sinu süsteemile sobivat käivitavat faili; lähtekoodi ega teise platvormi binaari ei saa nii lihtsalt käivitada.

LaTeX käsurealt

Eesmärk: `LaTeX`-i mõte on lihtne: lähtetekst jääb tekstifailiks, aga buildist sünnib väga täpselt juhitud PDF.

Põhikujud

- `pdflatex tere.tex` — kompileeri PDF
- `xelatex tere.tex` — parem Unicode tugi
- `latexmk -pdf tere.tex` — korda vajalikud käigud
- `latexmk -xelatex tere.tex` — `xelatex` automaatselt
- `ls tere.*` — vaata abifaile
- `open tere.pdf` — ava tulemus

Olulised failid

- `.tex` — lähtefail
- `.pdf` — tulemus
- `.aux` — ristviidete abi
- `.log` — kompileerimislogi

Pane tähele: Kui ristviited või sisukord ei ilmu kohe õigesti, ei tähenda see tingimata viga; LaTeX vajab sageli rohkem kui üht kompileerimiskäiku.

Lisad — täisspikker

Tase: **Referents**

Lisa A: kopeeritavad minitestid

Eesmärk: Kontrolli tulemust, mitte ainult käsu edukat käivitumist.

Harjutuse kontroll

- `ls kaust` — kas liigutatud fail on õiges kohas?
- `cat proov.txt` — kas vana ja lisatud rida on mõlemad alles?
- `sort loomad.txt | uniq -c` — kaks kassi, üks koer
- `python -m pip --version` — kas aktiivne Python kasutab projekti `.venv-i`?

Pane tähele: Kasuta uut katsekataloogi; näidete `>` suunamine kirjutab vana sisu üle.

Lisa B: spikrite register

Eesmärk: kõige lühemad meespead teemade kaupa: failid, tekst, võrk, arendus, loogika ja protsessid

Mida siit leiad

- Failid — liigu ja vaata
- Tekst — loe ja filtreeri
- Võrk — ühendu ja kopeeri
- Arendus — Git, Docker, koostamine
- Loogika — torud ja vead
- Protsessid — tööd ja signaalid

Pane tähele: See lisa on kõige lühem meespea; kui mõni käsk tundub hägune, mine tagasi vastava peatüki juurde.

Lisa C: sõnastik ja terminoloogia

Eesmärk: leia mõiste tähendus ning tunne ära ingliskeelne vaste dokumentatsioonis

Põhimõisted

- CLI — käsurida
- GUI — graafiline liides
- viip ehk prompt — ootab sisestust
- shell — tõlgendab käske
- lipp — käsu valik
- repo — Giti hoidla

Pane tähele: Täielik inglise-eesti terminoloogia on ka eraldi sõnastikulehel.

Lisa D: edasised võimalused ja lugejate soovid

Eesmärk: vali jätkuteema päris ülesande järgi ja saada konkreetne soovitus

Tugevad järgmised teemad

- R ja märkmikud (notebook) — analüüsi järgmine aste
- Git järgmine aste — harud ja konfliktid
- Shelliskriptid II — veakindlam loogika
- Statistika sild — andmetöö mõtteviis
- systemd timerid — ajastus Linuxis
- Varukoopiad — taastamine ja kontroll

Pane tähele: Tagasisides lisa peatüki link, kasutatud süsteem ja takerdumise koht; ära saada paroole ega privaatvõtmeid.

Lisa E: usaldusväärsed viited ja lisalugemine

Eesmärk: ametlikud käsiraamatud ja alustamisjuhendid shelli, SSH, Giti, Dockeri, JSON-i, SQLite'i ja LaTeX-i jaoks

Edasiõppimise teemad

- Shellid ja põhikäsud — bash, zsh, coreutils
- SSH ja võrk — OpenSSH, curl, wget
- Git ja GitHub — docs ja Pro Git
- Docker — get started ja compose
- JSON, CSV, XML, SQLite — ametlikud viited
- LaTeX — projekt ja CTAN

Pane tähele: Alusta iga teema puhul esimesest viitest; reference-manuaal on tavaliselt järgmine samm alles siis, kui põhikujutlus on olemas.

Lisa F: shelli seadistusfailid bash ja zsh jaoks

Eesmärk: Hoia üks baasprofiil oma shellile ja lisa ainult vajalikke aliaseid või funktsioone.

Käsud ja näited

- `ps -p $` — kontrolli praegu töötavat shelli
- `cp -i ~/.zshrc ~/.zshrc.varu` — tee olemasolevast failist varu; ära kirjuta varu pimesi üle
- `zsh -n ~/.zshrc` — kontrolli zsh süntaksit ilma käivitamata
- `bash -n ~/.bashrc` — kontrolli Bashi süntaksit ilma käivitamata
- `source ~/.zshrc` — käivita muudetud zsh seadistus praeguses shellis
- `zsh -f` — ava zsh kasutaja rc-faile lugemata
- `bash --noprofile --norc` — ava Bash profile lugemata

Märgid ja valikud

- `~/.zshrc` — interaktiivse zsh seadistus
- `~/.bashrc` — interaktiivse mitte-login Bashi seadistus
- `~/.bash_profile` — login-Bash; võib laadida `.bashrc`
- `PATH` — programmide otsinguteed; järjekord loeb

Pane tähele: Ära kirjuta olemasolevat profiili kogu näidisfailiga üle. `source` käivitab kõik failis olevad käsud.