ITBI Tutoriat 1

Albert Simon, Spiridon Mihnea-Andrei

October 15, 2025

1 Introducere

1.1 Ce facem la ITBI?

Ideea generala a cursului de ITBI este sa invatam cum sa lucram cu calculatorul si sa aflam despre diverse concepte si unelte esentiale in programare. Cursul de ITBI prefigureaza cursul de SO din anul 2 si se intersecteaza cu alte cursuri de anul 1 precum ASC. Cursul are anumite similaritati cu cel de USO de la sectia CTI.

1.2 Cum ne pregatim?

La ITBI vom lucra in general pe **Linux**. La laboratoare vom avea acces la Ubuntu de pe calculatoarele facultatii, insa pentru a lucra acasa si a exersa conceptele de la curs si laborator este util sa avem un mediu de lucru similar instalat local, pe calculatoarele personale. De asemenea, s-ar putea sa vrem la laboratoare sa lucram cu *sudo* ceea ce nu putem face de pe calculatoarele facultatii, fiindca implica drepturi de administrator. Ca sa facem rost de un astfel de mediu de lucru putem folosi o **masina virtuala** care simuleaza Linux-ul, sau sa **instalam direct pe calculator**.

Cel mai simplu este sa rulam dintr-o masina virtuala, insa poate rula mai incet si s-ar putea sa fie util sa avem Linux instalat pe calculator in general, pentru alte cursuri, nu doar pentru ITBI. Insa, daca ne instalam Linux pe calculator trebuie sa avem grija sa nu pierdem date, de asemenea ar fi bine sa facem **dual-boot** ca sa putem folosi in continuare si Windows. Mai jos aveti link-uri catre tutoriale ca sa va ajute sa va configurati calculatoarele.

- https://www.virtualbox.org/ – link descarcare VirtualBox
- https://ubuntu.com/download/desktop – link descarcare Ubuntu
- https://ubuntu.com/tutorials/install-ubuntu-desktop#1-overview – Ghid instalare Ubuntu pe calculator
- https://brb.nci.nih.gov/seqtools/installUbuntu.html – Ubuntu in masina virtuala
- https://merox.dev/blog/windows-11-ubuntu-25-04-dual-boot-guide/ – Ghid dual-boot cu Ubuntu

2 Recapitulare laboratoare

2.1 Laboratorul 1

Ne-am familizat cu ideea ca suntem un utilizator care foloseste un terminal pentru a da comenzi. Ar trebui sa retinem comenzile *id* (arata informatii despre user-ul cu care suntem logati), *man* (ne da documentatia la alte comenzi) si *pwd* (vom folosi cat de cat des ca sa aflam in ce director lucram.)

Deoarce la terminal nu avem acces la Notepad ca sa deschidem fisiere ar trebui sa retinem si diverse comenzi legate de manipularea fisierelor text. Pentru citire putem folosi *cat* ca sa afisam continutul unui fisier pe ecran. Daca este lung, i-l putem da unui pager precum *more* sau *less* ca sa ne plimbam prin text folosind arrow keys.

Ca sa editam un text din terminal folosim un editor text. Cel mai vechi/bine-cunoscut este *vi* insa putem incerca alte editoare foarte folosite precum *vim* sau *nano* care s-ar putea sa fie mai usor de folosit. E important sa alegeti ceva comod, nu va obliga nimeni, cel putin la acest curs, sa folositi ceva anume.

Vedeti si faimosul: https://stackoverflow.com/questions/11828270/how-do-i-exit-vim "How to exit Vim" de pe Stack Overflow. Alegeti ceva usor de folosit.

Recomandare: Aveti mai multa incredere in StackOverflow decat in ChatGPT.

2.2 Laboratorul 2

Am discutat despre cum sunt organizate fisierele intr-un calculator. Important este retinem ideea de **arbore cu radacina**, este universala ideea de structura arborescenta de directoare in orice sistem de operare. Foloseam comanda *tree* (s-ar putea sa trebuiasca sa fie instalata, *sudo apt install tree*) pentru a vedea un exemplu de astfel de structura. Retinem cateva comenzi ca sa stim sa navigam prin directoare si fisiere:

- *cd* schimbam directorul. Putem da o cale absoluta, relativa la unde ne aflam acum (retinem ca aflam cu *pwd*) sau putem scrie simplu *cd* ca sa ne intoarcem in $\sim$ (care este un simbol special, reprezentand home directory-ul utilizatorului nostru).
- *mkdir* creeaza directoare.
- *touch* creeaza fisiere (goale, implicit), la fel si daca dam *echo* intr-un fisier nou e.g. $echo\ Hello > new.txt$
- *rmdir* sterge directoare, dar functioneaza daca sunt goale (evitati celebrul $rm - rf$, asigurati-va ca nu aveti nimic ce nu trebuie sters inainte de *rmdir*)
- *cat* afiseaza continutul unui fisier text (putem folosi e.g. *hexdump* pentru alte tipuri de fisiere, in functie de natura lor), si cu *grep* cautam intr-un text. **Nu uitati:** avem pagerele *more* si *less* pentru a naviga in shell printr-un text lung.

In orice director din sistemul de fisiere Linux avem doua directoare speciale, anume . si .. unde directorul . reprezinta directorul curent si il folosim atunci cand vrem e.g. sa rulam un program *myprogram* din directorul curent, prin $./myprogram$, vom vedea imediat de ce nu merge sa scriem doar *myprogram*. Directorul .. reprezinta directorul anterior si il folosim cel mai des in *cd*, de exemplu printr-o comanda de genul $cd\ ../../$.

Exceptie: Cand suntem in radacina, directorul .. se refera tot la directorul curent, deoarece mai in spate de radacina arborelui nu putem sa mergem.

Tot in acest laborator am discutat ca shell-ul nostru nu este un simplu runner de comenzi, ci are si variabile. Aceste variabile se numesc **environment variables** si reprezinta un context in care ruleaza shell-ul nostru. In aplicatiile ulterioare vom folosi aceste variabile ale shell-ului si vom face script-uri cu ele in care le vom folosi chiar ca variabile din e.g. C++.

Variabilele din shell pot fi accesate cu simbolul $ inainte de numele lor. Le putem introduce in comenzi si se vor inlocui automat cu valoarea lor, e.g daca facem $echo\ \$PATH$ vom afla valoarea variabilei de mediu $PATH$. Daca vrem sa evitam acest comportament al shell-ului de a inlocui variabilele de mediu cu valoarea lor intr-o comanda, putem scrie $'\$PATH'$, single quotes determina folosirea literalmente a ceea ce se gaseste intre ele.

Observatie: Variabilele de mediu sunt toate siruri de caractere, intr-adevar comenzile pot si vor face intern o transformare din sir de caracter in intreg dar in momentul in care le folosim ele sunt siruri de caractere.

Exemplu: Chiar variabila $\$PATH$ este un exemplu bun. In momentul in care scriem primul cuvant intr-un shell, el intelege ca aceea este comanda, si o va cauta intr-o lista de directoare in variabila de mediu $\$PATH$. De aceea daca am compilat un program *myprogram* nu putem sa il rulam doar precizandu-i numele, trebuie sa-i scriem toata calea, **sau** sa il adaugam in $\$PATH$ deoarece acolo stie shell-ul sa caute. Ca sa aflam calea absoluta a unui program, folosim comanda *which*, de exemplu cand scriem *which echo*, intrebam shell-ul care *echo* stie sa il foloseasca.

Aceste variabile sunt specifice fiecarui shell in parte, deci daca deschid un alt terminal nu ma astept sa vad aceleasi variabile de mediu. De asemenea, daca modific o variabila de mediu si inchid terminalul, nu ar trebui sa ma astept sa o vad din nou cand mai deschid terminalul. Variabilele de mediu sunt specifice sesiunii si sunt setate (cateva dintre ele) dintr-un fisier care se executa inainte sa primiti terminalul si sa incepeti sa tastati comenzi.

Jucati-va putin intr-un shell: https://linuxsurvival.com/linux-tutorial-introduction/.