

Nume: _____ Prenume: _____

Județ: _____



Olimpiada de Informatică Aplicată AcadNet 2026

Calculatoare & Sisteme de Operare

Secțiunea 11–12

Citiți cu atenție toate subiectele înainte de a începe rezolvarea. Aveți 10 minute să adresați întrebări pentru clarificări înainte de a contoriza timpul de rezolvare.

Probele sunt independente, astfel că, după fiecare task (exercițiu sau subpunct) va trebui să atenționați unul dintre supraveghetori să vă puncteze. Pentru a obține punctajul pe un subpunct trebuie să arătați rezolvarea acestuia și să demonstrați că task-ul a fost realizat cu succes. Nu se vor acorda punctaje parțiale per subpuncte.

Pentru autentificarea pe mașina virtuală veți folosi <https://guacamole.grid.pub.ro>. Folosiți credențialele primite din partea supraveghetorilor. Folosiți ghidul guacamole primit din partea supraveghetorilor.

Nu aveți voie să modificați parolele pentru contul de administrator. În cazul modificării unei parole de administrator, sunteți descalificați.

Probele sunt punctate în funcție de dificultatea acestora.

Timp de lucru: 2 ore

Question	Points	Score
1	5	
2	5	
3	5	
4	10	
5	10	
6	10	
7	10	
8	10	
9	10	
10	10	
11	15	
Total:	100	

1. no-more-french-package (5p)

La această problemă va trebui să lucrați cu pachetul de limbă franceză. Și nu cel la care vă gândiți. Este vorba de **wfrench**, o listă cu toate cuvintele în limba franceză.

- (1 punct) Instalați pachetul **wfrench** pe sistem și listați pachetul de care acesta depinde.
- (1 punct) Localizați pe sistem calea absolută a fișierului **french** descărcat la subpunctul anterior.
- (2 puncte) Identificați cuvântul din fișierul **french** care îndeplinește simultan următoarele condiții:
 - Începe cu litera **f** și se termină cu litera **o**;
 - A doua literă este **a** sau **e**;
 - A treia literă este **r**;

- Conține exact **7** litere.

(d) (1 punct) Atât pentru astăzi. Timpul să faceți ce vă doriți cu adevărat: ștergeți pachetul de limbă franceză de pe sistem.

2. Automatizarea compilării cu Makefile (5p)

Se dorește gestionarea unui proiect C găsit în `~/LinkedList` format din modulele `linked_list.c` și `main.c`. Scrieți un fișier de tip **Makefile** care să implementeze următoarele funcționalități:

- (a) (1 punct) **Build:** Creați o regulă de build care să compileze fișierele sursă în fișiere obiect (`.o`), iar ulterior să le lege (*link-editare*) într-un executabil final numit `number_stat`. Asigurați-vă că procesul de compilare respectă ordinea logică a dependențelor.
- (b) (1 punct) **Clean:** Implementați o regulă `clean` care să șteargă executabilul rezultat și toate fișierele obiect generate în timpul procesului de build, pentru a lăsa directorul de lucru curat.
- (c) (1 punct) **Debug:** Adăugați o regulă numită `debug` care să refacă procesul de compilare, dar de data aceasta incluzând flag-ul de depanare (ex: `-g`) necesar pentru utilitare precum `gdb`.
- (d) (2 puncte) **Run:** Adăugați o regulă de `run` care să execute binarul `number_stat`. Regula trebuie să îndeplinească următoarele condiții:
 - Dacă executabilul nu există, acesta trebuie creat automat prin apelarea regulii de build înainte de rulare.
 - Trebuie să permită pasarea argumentelor din linia de comandă către executabil (de exemplu, folosind o variabilă precum `ARGS`).

3. Gestiunea concurenței prin Lockfiles (5p)

Atunci când modificați un fișier folosind un editor de text, acesta se asigură prin intermediul unui „lockfile” că fișierul nu va fi accesat simultan de o altă instanță a aplicației.

Scrieți un script Bash care să implementeze un mecanism similar de excludere mutuală (*mutual exclusion*), respectând următoarele cerințe:

- (a) (1 punct) Scriptul va primi ca argument în linia de comandă numele unui fișier. Dacă fișierul respectiv nu există pe disc, scriptul trebuie să îl creeze înainte de a continua execuția.
- (b) (2 puncte) Implementați mecanismul de blocare:
 - Înainte de a procesa fișierul, scriptul va verifica existența unui fișier de tip lock (ex: `nume_fisier.lock`).
 - Dacă lockfile-ul există, scriptul va afișa un mesaj de eroare și se va opri, semnalând că o altă instanță rulează deja.
 - Dacă lockfile-ul nu există, scriptul îl va crea și își va continua execuția.
- (c) (1 punct) Pentru a simula o procesare de lungă durată, scriptul nu își va finaliza execuția imediat (va rămâne activ într-o stare de așteptare sau buclă).
- (d) (1 punct) **Cleanup:** Asigurați-vă că, la finalizarea scriptului (chiar și în cazul unei întreruperi forțate, precum `Ctrl+C`), fișierul de tip lock este șters automat pentru a permite rulări ulterioare.

Hint: Pentru gestionarea întreruperilor, puteți utiliza comanda de tip `trap`.

4. Arhitectul de Pachete (Arch vs. Debian) (10p)

Viața pe Ubuntu este frumoasă, calmă și ușoară, dar nu toată lumea preferă această cale liniștită. Prietenul vostru Paul tocmai și-a instalat Arch Linux fără să dețină cunoștințe avansate, iar acum are nevoie de ajutorul vostru pentru conversia unor pachete software.

Scrieți un script Bash care, folosindu-se doar de utilitare standard de bază, primind ca argument un singur fișier de tip `.deb`, produce un fișier `.pkg.tar.zst` (formatul nativ Arch Linux).

Notă: Nu se admit soluțiile unde părți importante din rezolvare sunt delegate utilităților sau scripturilor externe, sau sunt împrumutate bucăți de cod din acestea.

- (a) (5 puncte) Construiți utilitarul astfel încât fișierul rezultat să respecte, odată instalat, arborescența de fișiere originală a pachetului Debian. În această etapă, ignorați metadatele și nu folosiți scripturi de tip *pre/post install*.
- (b) (2 puncte) Extindeți scriptul pentru a importa elementele de metadate din fișierul de control Debian (ex: nume, versiune, descriere, arhitectură). Vor fi migrate toate câmpurile relevante, cu excepția celor legate de dependențe. Rezultatul trebuie să conțină fișierul de metadate specific Arch (`.PKGINFO`).
- (c) (3 puncte) Finalizați scriptul asigurând migrarea scripturilor de instalare. Dacă pachetul Debian conține scripturi de tip `preinst`, `postinst`, `prerm` sau `postrm`, acestea trebuie convertite într-un fișier de instalare compatibil cu managerul de pachete `pacman` (formatul `.INSTALL`).

Hint: Folosiți utilitare precum `ar` sau `tar` pentru a dezarhiva pachetul Debian și `zstd` pentru compresia finală.

5. Integritate și Socket-uri (10p)

- (a) (1 punct) Generați o pereche de chei GPG pentru utilizatorul `student` cu adresa de email `student@upb.ro`, fără a seta o parolă (non-interactive/batch mode). Apoi extrageți informațiile despre procesorul sistemului gazdă și salvați acest output într-un fișier numit `cpu_info.txt`.
- (b) (2 puncte) Semnați fișierul `cpu_info.txt` folosind cheia GPG creată anterior. Generați o semnătură de tip *detached* (semnătură separată) și salvați-o în fișierul `cpu_info.sig`.
- (c) (4 puncte) Configurați un mecanism de transfer folosind **Unix Domain Sockets**:
 - Creați un socket localizat la calea `/tmp/transfer.sock`.
 - Folosiți utilitarul `nc` (netcat) pentru a porni un proces de tip server care să asculte pe acest socket și să redirecționeze datele primite într-o arhivă numită `received.tar`.
 - Dintr-un director diferit, folosiți un al doilea proces `nc` pentru a transmite către socket o arhivă `.tar` care să conțină fișierele `cpu_info.txt` și `cpu_info.sig`.
- (d) (3 puncte) Realizați un script Bash care să automatizeze validarea integrității. Scriptul trebuie să:
 - Extragă fișierele primite.
 - Verifice dacă fișierul `cpu_info.txt` corespunde semnăturii `cpu_info.sig` folosind `gpg -verify`.
 - Afișeze mesajul **"Validare Reusita"** dacă verificarea este confirmată sau **"Eroare Integritate"** în caz contrar.

6. Compatibilitate (10p)

După ce Academia Nord-Est Tecuci a actualizat serverul lor de compilări au observat că au apărut pachete care nu mai puteau fi construite.

Investigând problema aceasta, echipa internă a realizat că pachetul `automake` a avut schimbări ce provoacă stricări în versiunea 1.17. Pentru a putea păstra schimbările de securitate dintre versiunea aceasta și versiunea cea mai nouă, ei au publicat un fișier `patch` de compatibilitate.

Tu, fiind precaut, ai găsit acest fișier cât timp verificai ce probleme ar putea apărea înainte să actualizezi serverele politehnicii.

- (a) (1 punct) Descarcă folosind git cel mai recent cod sursă de la proiectul GNU Automake.
- (b) (2 puncte) Folosește git pentru a aduce codul înapoi la versiunea `v1.16.4`.
- (c) (3 puncte) Aplică patch-ul `~/NTRU/compatibilitate_automake.patch`.
- (d) (3 puncte) Compilează și instalează pachetul în prefixul `~/NTRU/am/`.
- (e) (1 punct) Printează versiunea binarului rezultat.

7. Acadcore (10p)

Tot lovindu-te de problemele care apar când folosești containere pentru a orchestra totul, te decizi că ai avut destul și vrei să te întorci la mașini virtuale.

Pentru a-ți îndeplini scopul, te gândești să te folosești de ultimul `.iso` pe care l-ai folosit înainte de revoluția k8s, dar rapid observi o problema.

Discul este partiționat complet incorect, iar formatarea partițiilor este greșită!

Dorind să arăți superioritatea mașinilor virtuale prietenului tău Lucian, ramâi neclintit și rezolvi imaginea astfel:

- (a) (2 puncte) Copiază `~/Tc/tc.iso` într-un fișier nou și repartizionează-l astfel: Prima partiție trebuie marită până la 80MB (fără a o șterge) și declarată bootabilă. Spațiul rămas apoi trebuie umplut cu încă o partiție primară.
- (b) (3 puncte) Reformatează cea de-a doua partiție cu sistemul de fișiere `ext4`.
- (c) (3 puncte) Creează un fișier normal în cea de a doua partiție numit `hydra.txt`
- (d) (2 puncte) Pornește o mașină virtuală pe arhitectura `x86_64`, fără display grafic, cu harddiskul fiind imaginea proaspăt creată.

Hint: Pentru a ieși din consola serial QEMU trebuie apasat `<Ctrl-a>` `x`

8. Turnul Babel (10p)

Așezându-vă pentru prima oară la tastatura unui calculator dintr-un laborator de-al Politehnicii București, în calitate de proaspăt student, vă lovește o realizare cumplită: acesta este în limba engleză, pe care nu o cunoașteți!

Fiind un calculator de laborator, evident, nu aveți acces la niciun fel de escaladare de privilegii (nu veți folosi pe mașina gazdă sub nicio formă comenzi precum `sudo`, `su`, `doas`, sau similare), iar laborantul refuză vehement să vă ofere parola contului de administrator. Din fericire, sunteți priceput la haiducie digitală și constatați că utilizatorul dumneavoastră, neprivilegiat, se află în grupul `docker`.

Notă: Scopul este modificarea fișierelor de configurare de pe gazdă folosind mediul izolat oferit de container pentru a bypass-a restricțiile de permisiuni.

- (a) (1 punct) Lansați un container docker, folosind un utilizator neprivilegiat.
- (b) (6 puncte) Obțineți acces privilegiat la mașina gazdă, prin orice mijloace necesare.

- (c) (2 puncte) Schimbați configurarea regională (`locales`) de pe mașina gazdă.
- (d) (1 punct) Explicați de ce apartenența la grupul `docker` este echivalentă cu privilegii depline pe gazdă.

9. Configurare Rețea și Servicii (10p)

Se consideră următoarea topologie de rețea:

`node1 (eth0) <-> router <-> node2 (eth0)`

Pentru deplasarea între noduri, se vor folosi comenzile `go node1`, `go node2` și `go router`.

- (a) (2 puncte) **Adresare IP:** Configurați adresa IPv4 `12.45.21.50/24` pe interfața `eth0` a nodului `node1` și adresa IPv4 `12.45.22.50/24` pe interfața `eth0` a nodului `node2`. Asigurați-vă că interfețele sunt activate (*up*).
- (b) (3 puncte) **Routare:** Configurați gateway-ul implicit (*default gateway*) pe ambele noduri pentru a permite comunicarea prin intermediul routerului:
 - `node1` va folosi adresa `12.45.21.254` drept gateway.
 - `node2` va folosi adresa `12.45.22.254` drept gateway.
- (c) (2 puncte) **Rezoluție de Nume (Local Hostname Resolution):** Pe fiecare nod (`node1` și `node2`), configurați fișierul corespunzător astfel încât să fie posibile următoarele acțiuni:
 - De pe `node1`: `ping node2`, `ping router-node1-intf` și `ping router-node2-intf`.
 - De pe `node2`: `ping node1`, `ping router-node1-intf` și `ping router-node2-intf`.

Notă: Utilizați adresele IP ale interfețelor routerului corespunzătoare gateway-urilor configurate anterior.

- (d) (3 puncte) **Acces SSH fără parolă:** Realizați configurațiile necesare pentru a permite accesul SSH bazat pe chei între utilizatori, respectând următoarele condiții:
 - De pe `node1`, utilizatorul curent trebuie să se poată conecta la `mihaela@node2` fără parolă.
 - De pe `node2`, utilizatorul curent trebuie să se poată conecta la `flavia@node1` fără parolă.
 - Parola curentă a utilizatorilor ("`acadnet`") nu trebuie ștearsă sau modificată.

10. cgroupv2 (10p)

Într-o bună zi ajungi la muncă și observi că serverul de producție era cu totul îngenunchiat. Investigând, ți-ai dat seama că totul a fost cauzat de către un coleg ce rula un program matematic care acapara toate resursele de CPU și IO ale sistemului.

Deoarece acesta nu cooperează în a scrie cod ce permite și altor procese să coexiste pe server, ai decis să îi pui niște bețe în roate folosind interfața kernelului `cgroupv2`.

- (a) (2 puncte) Crează un `cgroup` nou cu numele `limiter` și adaugă procesul `collatzProver` în el.
- (b) (2 puncte) Găsește identificatorul "major:minor" pentru `/dev/sda` printr-un oneliner.
- (c) (3 puncte) Limitează viteza maximă de scriere pentru `/dev/sda` să fie `2048Bps`, iar timpul maxim de CPU folosit să fie `110ms` pentru fiecare `100ms`.
- (d) (3 puncte) Pentru a te asigura că schimbările tale sunt permanente, creează un script în `/usr/local/bin` cu numele `collatzProver` (pentru a putea acapara numele în shell), ce pornește procesul și automatizează toți pașii de mai sus.

11. Fantoma în Namespace (15p)

Fiind obsedat de securitate și izolare la maxim, un programator dorește să pună pe picioare un container de webserver. Noutatea, însă, este că acest container nu este deschis de `root`, ci de un utilizator normal! Numim acestea containere **rootless**.

Acesta are nevoie de acces la directorul `~/Podman/data` care conține `index.html` și două fișiere de log, un setup tipic pentru un webserver Apache. Încercând să pornească containerul folosind `podman-compose up`, acesta crapă imediat raportând că nu are acces la fișierele din `~/Podman/data`. Este perplexant, deoarece utilizatorul curent are acces și este owner-ul acestor fișiere!

Fără să modifice permisiunile directorului `~/Podman/data` (se va lăsa 755, **NU** 777) și fără să modifice conținutul vreunui fișier, identifică de ce containerul pornit de utilizatorul `student` nu are acces la aceste fișiere și rezolvă problema.

Notă: Când închizi containerul sau dacă acesta se închide singur (crash), rulează mereu `podman-compose down` pentru a șterge fișierele reziduale și a evita erorile la repornire.

Hint: Verifică `/etc/subuid` și instrucțiunile din `Dockerfile`.

- (a) (11 puncte) Rezolvă problema de permisiune din container fără să modifice permisiunile prin `chmod` și demonstrează că funcționează.
- (b) (4 puncte) Explică care este motivul tehnic (legat de user namespaces) pentru care containerul nu a avut inițial acces la acele fișiere, deși utilizatorul de pe host era owner-ul lor.