Test practic la SO - varianta nr. 2

Realizați o aplicație formată din următoarele trei componente, definite în continuare, care vor fi dispuse conform ierarhiei de mai jos:

```
.
├── main.sh
├── tools
│   └── filetypeandids.c
    └── call-ID.sh
```

1. Să se scrie un program C, numit "filetypeandids.c", conform specificației următoare:

Programul va primi un argument în linia de comandă; în caz contrar se va termina cu codul de eroare 10. Programul va considera acel argument ca fiind un nume de fișier și va determina, folosind o funcție din API-ul POSIX, tipul de fișier, precum și ID-ul userului proprietar și ID-ul grupului proprietar. Dacă sunt erori la apelul acelei funcții, atunci programul își va încheia execuția cu codul de terminare 11. Altfel, programul va afișa pe ieșirea stdout, folosind o funcție din biblioteca stdio, un mesaj format din numele fișierului și două numere întregi reprezentând ID-urile specificate mai sus, separate prin caracterul ':', și terminat cu caracterul '\n'.
Apoi programul își va încheia execuția cu codul de terminare 0, dacă tipul fișierului este fișier normal, respectiv cu codul de terminare 1, în caz contrar.

2. Să se scrie un script bash, numit "call-ID.sh", conform specificației următoare:

Scriptul va primi un argument în linia de comandă; în caz contrar se va termina cu codul de eroare 2.
Scriptul va considera acel argument ca fiind un nume de fișier și va apela executabilul "filetypeandids" (obținut prin compilarea sursei "filetypeandids.c"), transmițându-i acestuia ca argument în linia de comandă, acel nume de fișier pe care l-a primit el în linia de comandă.
Apoi scriptul va decide, pe baza codului de terminare a execuției programului "filetypeandids", dacă numele de fișier primit este un fișier de tip normal sau nu. Dacă fișierul este normal, atunci i se vor dezactiva permisiunile de (w)rite și e(x)ecute pentru (o)thers.
Iar dacă fișierul primit ca și parametru nu este fișier normal, se va verifica dacă acesta este un director și, în caz afirmativ, va efectua următoarea procesare a conținutului acelui director: va parcurge, folosind o structură repetitivă, toate intrările directe din acel director și, pentru fiecare intrare, **scriptul se va apela recursiv pe sine însuși**, cu acea intrare ca parametru în linia de comandă. (Atenție: așadar, aici trebuie să implementați o recursie explicită!)

3. Să se scrie un alt script bash, numit "main.sh", conform specificației următoare:

Scriptul va primi un argument în linia de comandă și, mai întâi, va face următoarele verificări, în ordinea următoare:
i) va verifica că în directorul în care se află "main.sh", există un subdirector numit "tools" ce conține scriptul "call-ID.sh" și că are permisiunea de execuție a acestuia, iar în caz negativ va afișa pe **ieșirea stdout** un mesaj de eroare adecvat și se va termina cu codul 10;
ii) va verifica că în acel subdirector numit "tools" se află și programul "filetypeandids.c" și că are permisiunea de citire a acestuia, iar în caz negativ va afișa pe **ieșirea stdout** un mesaj de eroare adecvat și se va termina cu codul 11;
iii) va verifica că în același subdirector "tools" se mai află și un fișier numit "filetypeandids" (executabilul obținut prin compilare din programul "filetypeandids.c"), iar în caz negativ va apela compilatorul gcc pentru a-l produce și, doar dacă vor fi erori la compilare, va afișa pe **ieșirea stdout** un mesaj de eroare adecvat și se va termina cu codul 12;
iv) va verifica dacă argumentul primit reprezintă un director existent și asupra căruia are permisiune de citire, iar în caz negativ va afișa pe **ieșirea stdout** un mesaj de eroare adecvat și se va termina cu codul 13.

Apoi scriptul "main.sh" va invoca scriptul "call-ID.sh" din acel subdirector "tools" specificat mai sus, transmițându-i acestuia ca argument în linia de comandă, argumentul pe care l-a primit el în linia de comandă.
Invocarea scriptului "call-ID.sh" se va face printr-o comandă compusă de tip pipeline, care să proceseze outputul produs prin execuția scriptului "call-ID.sh" în maniera următoare: se vor păstra doar acele intrări ce au ca și user ID valoarea ID-ului utilizatorului curent, apoi se vor selecta doar numele fișierului și ID-ul grupului proprietar, înlocuind caracterul separator “:” cu caracterul “*” în perechile de forma "numefișier:group-ID" rezultate în urma procesării.
La finalul procesării pipeline-ului, scriptul "main.sh" se va termina cu codul de terminare 0.

Observație: pentru o testare corectă, asigurați-vă că rulați scriptul “main.sh” pe un director ce conține cel puțin un fișier ce are ca și owner, userul vostru. Puteți testa și pe directorul / , doar ca timpul de execuție va crește.

Barem de corectare

Observații:
- programul C și cele două scripturi trebuie plasate într-o ierarhie de directoare conform celor descrise în enunțul problemei (și apelate în mod corespunzător poziției lor în ierarhia respectivă).
- conform specificației date, programul C poate folosi biblioteca standard doar pentru afișarea rezultatelor; interacțiunea cu fișierul (aflarea tipului, proprietarului, etc.) se va face folosind doar API-ul POSIX.
- pentru implementarea parcurgerii recursive se va respecta specificația dată pentru scriptul "call-ID.sh".
- **dacă nu respectați toate condițiile precedente** (de exemplu, dacă plasați vreunul dintre cele două fișiere apelate de “main.sh” într-un alt director, sau dacă folosiți comenzi precum find sau ls -R pentru parcurgerea recursivă), atunci vom evalua corectitudinea rezolvării, dar **punctajul total acordat va fi înjumătățit**.
- fiecare punctaj din barem se acordă integral, sau deloc.

1. Baremul pentru programul "filetypeandids.c":
a) 2p - implementarea corectă, conform specificației date, a determinării tipului, ID-ului utilizatorului proprietar și ID-ului grupului proprietar al fișierului.
b) 1p - afișarea rezultatului pe ieșirea stdout, conform specificației date.
c) 1p - tratarea tuturor erorilor posibile, conform specificației date.
d) 1p - programul se compilează fără erori și fără warning-uri.
Total: 5p

2. Baremul pentru scriptul "call-ID.sh":
a) 1p - invocarea corectă a programului executabil "filetypeandids", conform specificației date.
b) 1p - luarea deciziei de procesare, dacă argumentul primit este un fișier normal, conform specificației date.
c) 2p - iar dacă argumentul primit este un director, parcurgerea corectă a intrărilor directe din directorul respectiv și apelarea recursivă a scriptului pentru fiecare intrare din acesta, conform specificației date.
d) 1p - scriptul nu are erori de sintaxă la execuție.
Total: 5p

3. Baremul pentru scriptul "main.sh":
a) 1p - implementarea corectă, conform specificației date, a verificării descrise la punctul i) din specificație.
b) 1p - implementarea corectă, conform specificației date, a verificării descrise la punctul ii) din specificație.
c) 1p - implementarea corectă, conform specificației date, a verificării descrise la punctul iii) din specificație.
d) 1p - implementarea corectă, conform specificației date, a verificării descrise la punctul iv) din specificație.
e) 1p - invocarea corectă a scriptul "call-ID.sh", în cadrul pipeline-ului, conform specificației date.
f) 1p - apelul comenzii potrivite, în cadrul pipeline-ului, pentru a păstra doar liniile de text ce au ca și user ID, ID-ul utilizatorului curent ce rulează scriptul.
g) 1p - apelul comenzii potrivite, în cadrul pipeline-ului, pentru a păstra doar numele fișierului și ID-ul grupului proprietar.
h) 1p - apelul comenzii potrivite, în cadrul pipeline-ului, pentru înlocuirea caracterului separator “:” cu “*” , conform specificației date.
i) 1p - scrierea corectă a întregului pipeline pentru procesarea descrisă în specificația dată.
j) 1p - scriptul nu are erori de sintaxă la execuție.
Total: 10p