Test practic la SO – varianta nr. 2

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

```
.
├── main.sh
├── programs
│   └── build_scattered_number.c
└── scripts
    ├── listdir.sh
    └── pipeline.sh
```

1. Să se scrie un program C, numit "build_scattered_number.c", conform specificației următoare:
Programul va primi un argument în linia de comandă; în caz contrar se va termina cu codul de eroare 1. Programul va considera acel argument ca fiind o cale către un fișier și în cazul în care nu are drepturi de citire asupra sa, va afișa un mesaj corespunzător pe stderr și se va termina cu codul de eroare 1.
Folosind doar apeluri din cadrul API-ului POSIX, programul va construi un număr cu toate cifrele, în ordinea în care apar, care se află în ultimele 8 caractere ale fișierului (e.g., pentru “x+y-1ab2”, se va construi numărul “12”). În cazul în care fișierul are mai puțin de 8 caractere, practic se va citi tot fișierul, iar dacă nu sunt deloc cifre, numărul construit va fi considerat 0.
Dacă sunt erori la apelurile din cadrul funcțiilor din API-ul POSIX se va afișa un mesaj corespunzător și se va ieși din program cu codul de exit 2. Altfel, programul va scrie pe stdout calea către fișier primită ca și argument și numărul construit, separate de caracterul “:” și urmate de caracterul “\n”, și se va termina cu codul de terminare 0.

2. Să se scrie un script bash, numit "main.sh", conform specificației următoare:
Scriptul va primi un argument în linia de comandă și va face următoarele verificări, în ordinea următoare:
i) va verifica dacă a primit un singur argument la linia de comandă, iar în caz contrar va afișa un mesaj corespunzător și se va termina cu codul 1.
ii) va verifica dacă în directorul în care se află "main.sh", există un subdirector numit "scripts" ce conține scripturile "pipeline.sh" și “listdir.sh”, iar în caz negativ va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul 2.
iii) va verifica, de asemenea, dacă are drept de execuție pentru cele două scripturi din directorul "scripts", iar în caz contrar li se va adăuga drept de execuție doar pentru proprietar.
iv) va verifica dacă în directorul în care se află "main.sh", există un subdirector numit "programs" ce conține un program numit "build_scattered_number.c" și că are permisiunea de citire a acestuia, iar în caz negativ va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul 3.
v) va verifica dacă în același director ca și "build_scattered_number.c" există "build_scattered_number" iar în caz negativ se va compila programul C cu denumirea "build_scattered_number" pentru executabil. În cazul unei erori la compilare, se va afișa un mesaj corespunzător la stderr și se va termina cu codul 4.

Apoi scriptul "main.sh" va invoca scriptul "pipeline.sh", transmițându-i în linia de comandă, argumentul pe care l-a primit el în linia de comandă, și la final va afișa codul de terminare a execuției acestuia.

3. Să se scrie un al doilea script bash, numit "pipeline.sh", conform specificației următoare:
Scriptul va primi un argument în linia de comandă; în caz contrar va afișa un mesaj corespunzător și se va termina cu codul 1. Apoi scriptul "pipeline.sh" va invoca scriptul "listdir.sh" (aflat în același director cu el), transmițându-i în linia de comandă, argumentul pe care l-a primit el în linia de comandă.
Invocarea scriptului "listdir.sh" se va face printr-o comandă compusă de tip pipeline, care să proceseze outputul produs prin execuția scriptului "listdir.sh" în maniera următoare: se vor obține primele 5 fișiere cu cele mai mari numere, pentru care se va afișa doar calea acestora, fără alte extra informații.

4. Să se scrie un al treilea script bash, numit "listdir.sh", conform specificației următoare:
Scriptul va primi un argument în linia de comandă și va face următoarele verificări, în ordinea următoare:
i) va verifica daca a primit un argument la linia de comandă și că acesta este un director; în caz negativ, se va afișa un mesaj corespunzător pe stderr și se va termina cu codul 1.

ii) va verifica dacă în directorul "programs" exista fișierul "build_scattered_number" și are drept de execuție asupra sa; în caz negativ, se va afișa un mesaj corespunzător pe stderr și se va termina cu codul 2.

Apoi scriptul "listdir.sh", va implementa o recursie **explicită**, printr-o funcție recursivă, pentru a itera prin toate fișierele din cadrul directorului primit ca și argument. Numai pentru fișierele care au extensia ".txt", se va apela executabilul "build_scattered_number" împreună cu calea către fișierul respectiv. În cazul în care apelul programului executabil se va termina cu un cod de exit nenul, se va scrie pe stderr un mesaj corespunzător care să conțină codul respectiv și numele fișierului ".txt" procesat de acel apel, și se va continua execuția normală, cu următoarea iterație. (Atenție: așadar, aici trebuie să implementați o recursie explicită!)

Barem de corectare

Observații:
- programul C și cele trei 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 de C doar pentru afișarea rezultatelor / erorilor; interacțiunea cu fișierul se va face folosind doar API-ul POSIX.
- pentru implementarea parcurgerii recursive se va respecta specificația dată pentru scriptul "listdir.sh".
- **dacă nu respectați toate condițiile precedente** (de exemplu, dacă plasați vreunul dintre cele trei fișiere apelate pornind de la “main.sh” într-un alt director, sau dacă folosiți comenzi precum find sau ls -R pentru parcurgerea recursivă în cadrul "listdir.sh"), 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 "build_scattered_number.c":
a) 0.5p + 0.5p – testarea numărului de argumente și, respectiv, a dreptului de citire asupra fișierului.
b) 1.5p + 0.5p – construirea numărului și afișarea rezultatului pe ieșirea stdout, conform specificației date, inclusiv citirea corectă în cazul excepțional când fișierul conține informație insuficientă.
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 "main.sh":
a) 0.5p – implementarea corectă, conform specificației date, a verificării descrise la punctul i) din specificație.
b) 3 x 0.5p – implementarea corectă, conform specificației date, a celor 3 verificări descrise la punctul ii) din specificație.
c) 2 x 0.5p – implementarea corectă, conform specificației date, a verificărilor descrise la punctul iii) din specificație.
d) 2 x 0.5p – implementarea corectă, conform specificației date, a verificărilor descrise la punctul iv) din specificație.
e) 1p – implementarea corectă, conform specificației date, a punctului v) din specificație.
f) 0.5p + 0.5p – invocarea corectă a scriptului "pipeline.sh" și afișarea codului de exit, conform specificației date.
Total: 6p

3. Baremul pentru scriptul "pipeline.sh":
a) 0.5p – implementarea corectă, conform specificației date, a verificării primirii unui argument.
b) 0.5p – invocarea corectă a scriptului "listdir.sh", în cadrul pipeline-ului, conform specificației date.
c) 1p – apelul comenzii potrivite, în cadrul pipeline-ului, pentru ordonarea numerelor, conform specificației date.
d) 1p – apelul comenzii potrivite, în cadrul pipeline-ului, pentru a obține primele 5 fișiere, conform specificației date.
e) 1p – apelul comenzii potrivite, în cadrul pipeline-ului, pentru afișarea doar a căii către fișiere, fără alte extra informații, conform specificației date.
Total: 4p

4. Baremul pentru scriptul "listdir.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 a funcției recursive de parcurgere a unui director, plus:
c_1) 0.5p – implementarea corectă a testării pentru extensia “.txt”.

c_2) 0.5p – apelarea corectă a programului “build_scattered_number”.
c_3) 0.5p – gestionarea conform specificației a cazului în care apelul programului “build_scattered_number” se termină cu un cod de eroare, conform specificației date.
d) 0.5p – apelarea funcției de parcurgere pentru argumentul primit.
Total: 5p