Test practic la SO – varianta nr. 3

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
│   └── file_pages.c
└── scripts
    ├── listdir.sh
    └── pipeline.sh
```

1. Să se scrie un program C, numit "file_pages.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 un nume de fișier (în care nu apare caracterul '+') și va determina, folosing apeluri din API-ul POSIX, următoarele informații: i) numărul de pagini care ar fi ocupate în memorie dacă tot fișierul ar fi citit deodată. Dimensiunea unei pagini este considerată a fi 4096 octeți. (Exemplificări: 0 octeți => 0 pagini ; 1 octet => 1 pagină ; 1250 octeți => 1 pagină ; 4096 octeți => 1 pagină ; 4097 octeți => 2 pagini ; ș.a.m.d.) ; ii) valoarea octetului aflat în fișier la poziția egală cu numărul de pagini calculat la punctul i).
În caz de eroare la oricare dintre apelurile POSIX utilizate, programul își va încheia execuția cu un cod de terminare unic, nenul. Altfel, programul va afișa pe stdout un mesaj compus din: numărul de pagini, urmat de separatorul "+++", urmat de argumentul primit, urmat de "+++", urmat de valoarea octetului citit, urmat de un caracter newline. (Exemplu de output posibil: `2+++programs/file_pages.c+++95`)

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 în directorul în care se află "main.sh", există un subdirector numit "scripts" ce conține un script numit "pipeline.sh" și că are permisiunea de execuție a acestuia, iar în caz negativ va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul 1;
ii) va verifica dacă în directorul în care se află "main.sh", există un subdirector numit "programs" ce conține un program numit "file_pages.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 2;
iii) va verifica dacă argumentul primit reprezintă un director existent și asupra căruia are permisiune de citire, iar în caz negativ va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul 3.
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 se va termina cu codul de exit 2.
Mai întâi scriptul va verifica dacă în subdirectorul "scripts" există un script numit "listdir.sh" și că are permisiunea de execuție a acestuia, iar în caz negativ va afișa pe ieșirea stderr un mesaj de eroare adecvat și se va termina cu codul 1.
Apoi scriptul "pipeline.sh" va invoca scriptul "listdir.sh", 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 va înlocui caracterul separatorul "+++" cu stringul " ---> ", iar apoi se vor afișa pe stdout doar numele fișierului și octetul citit din acesta, dar fără numărul de pagini calculat. De asemenea, scriptul va calcula suma numerelor de pagini ale tuturor fișierelor procesate fără erori și o va afișa pe stderr. Exemplu posibil de rezultate afișate pe ecran:

```
./main.sh ---> 65
./programs/file_pages ---> 251
./programs/file_pages.c ---> 97
./scripts/listdir.sh ---> 101
./scripts/pipeline.sh ---> 75
Total number of pages: 8
```

4. Să se scrie un al treilea script bash, numit "listdir.sh", conform specificației următoare:
Scriptul va primi un argument în linia de comandă; în caz contrar se va termina cu codul de exit 2.
Mai întâi scriptul va verifica dacă în subdirectorul "programs" se mai află și un fișier numit "file_pages" (executabilul obținut prin compilare din programul "file_pages.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 stderr un mesaj de eroare adecvat și se va termina cu codul de exit 3.
Dacă argumentul primit este un fisier normal, scriptul "listdir.sh" va apela executabilul "file_pages", transmițându-i acestuia ca argument în linia de comandă, argumentul pe care l-a primit el în linia de comandă.
Iar dacă argumentul este un director, scriptul "listdir.sh" 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 transmisă ca parametru. (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; interacțiunea cu fișierul (aflarea tipului, lungimii, etc.) 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), 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 "file_pages.c":
a) 0.5p + 0.5p – implementarea corectă, conform specificației date, a determinării numărului de pagini.
a) 3 x 0.5p – implementarea corectă, conform specificației date, a determinării valorii octetului specificat.
b) 0.5p – 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 "main.sh":
a) 0.5p + 0.5p – implementarea corectă, conform specificației date, a verificărilor descrise la punctul i) din specificație.
b) 0.5p + 0.5p – implementarea corectă, conform specificației date, a verificărilor descrise la punctul ii) din specificație.
c) 0.5p + 0.5p – implementarea corectă, conform specificației date, a verificărilor descrise la punctul iii) din specificație.
d) 0.5p + 0.5p – invocarea corectă a scriptului "pipeline.sh" și, respectiv, afișarea codului de exit, conform specificației date.
Total: 4p

3. Baremul pentru scriptul "pipeline.sh":
a) 0.5p – implementarea corectă, conform specificației date, a verificării referitoare la argumentul primit.
b) 1p – implementarea corectă, conform specificației date, a verificării referitoare la scriptul "listdir.sh".
c) 1p – invocarea corectă a scriptului "listdir.sh", în cadrul pipeline-ului, conform specificației date.
d) 1p – apelul comenzii potrivite, în cadrul pipeline-ului, pentru înlocuirea stringului separator '+++' cu stringul " ---> ", conform specificației date.
e) 1.5p + 0.5p – apelul comenzilor potrivite, pentru a calcula și, respectiv, afișa numărul total de pagini, conform specificației date.
Total: 5.5p

4. Baremul pentru scriptul "listdir.sh":
a) 0.5p – implementarea corectă, conform specificației date, a verificării referitoare la argumentul primit.
b) 0.5p + 1p – implementarea corectă, conform specificației date, a verificării referitoare la programul executabil "file_pages" și apelul compilatorului, dacă este cazul, conform specificației date.
c) 1p – invocarea corectă a programului executabil "file_pages", conform specificației date.
d) 1.5p – luarea deciziei de procesare, dacă argumentul primit este un director, prin parcurgerea iterativă a intrărilor directe din acel director...
e) 1p – ... și apelarea recursivă a scriptului pentru fiecare intrare din acesta, conform specificației date.
Total: 5.5p