PROGRAMAREA CALCULATOARELOR
Tema 3 - Fișiere, Stringuri și Structuri

Termen de predare: 13-12-2019 23:55(**deadline hard**)

Responsabili Temă: **Ștefania Ghiță, Vlad Nicolaescu**

Contents

Descrierea problemei

Starman preia mai multe imagini din spațiu pe care vrea să le transmită pe Pământ. Pentru ca transmiterea să fie eficientă, acesta vrea mai întâi să grupeze imaginile într-o arhivă, astfel încât să trimită un singur fișier. Însă Starman nu are acces la un utilitar pentru arhivare. Scopul nostru este să implementăm un program de arhivare, ca să îl ajutăm pe Starman să trimită cât mai multe poze.

Implementați un program în limbajul C care simulează un utilitar ce creează o arhivă de tipul tar. Utilitarul este reprezentat de un fișier executabil cu numele **archiver**, care atunci când rulează poate interpreta anumite comenzi de la **standard input** (**stdin**). Comenzile pe care programul poate să le interpreteze sunt:

- `create nume_arhivă nume_director` - creează o arhivă tar cu numele *nume_arhivă* ce conține fișiere din directorul *nume_director*.
- `list nume_arhivă` - afișează numele fișierelor conținute în arhiva *nume_arhivă*.
- `extract nume_fișier nume_arhivă` - extrage conținutul fișierului *nume_fișier* salvat în arhiva *nume_arhivă*.
- `exit` - încheie execuția programului.

Cerința 0 - archiver

Scrieți un program în limbajul C care primește comenzi de la **standard input** (**stdin**) pe care le interpretează astfel:

- Dacă primește comanda **exit** va încheia execuția programului;
- Dacă primește o comandă cunoscută (**create**, **list** sau **extract**) și implementată va executa codul asociat acelei comenzi și apoi va aștepta un nou input;
- Dacă primește o comandă necunoscută (nu este una dintre **create**, **list**, **extract** sau **exit**), greșită (nu are numărul de parametri așteptat) sau neimplementată, va afișa textul "> *Wrong command!*" pe o linie nouă și va aștepta un nou input.

Fiecare comandă reprezintă o linie. Puteți presupune că o linie are maximum 511 caractere.

Un posibil scenariu de execuție al temei este în exemplul de mai jos:

```
user@user-pc:~\$ ./archiver
create arhiva_mea.tar
> Wrong command!
create arhiva_mea.tar imagini/
> Done!
list arhiva_mea.tar
> imagine1.png
> imagine2.png
> imagine3.png
exit
user@user-pc:~\$
```

Cerința 1 - create

Adăugați programului scris anterior funcționalitatea asociată comenzii **create**. Sintaxa acestei comenzi este:

```
create nume_arhiva nume_director
```

Această comandă creează o arhivă tar cu fișiere din directorul *nume_director*, pe baza informațiilor extrase din două fișiere auxiliare existente, **usermap.txt** și **files.txt**.

- Parametrul **nume_arhivă** reprezintă un exemplu de nume pe care utilizatorul îl poate da arhivei pe care vrea să o creeze.
- Parametrul **nume_director** reprezintă directorul în care se află fișierele care vor fi incluse în arhivă. Acest parametru este de fapt calea relativă a directorului față de directorul în care se află executabilul *archiver* (exemplu: dacă executabilul se află în /home/user/tema3/, iar fișierele sunt în /home/user/tema3/imagini/, atunci parametrul va avea valoarea imagini/). Se garantează că acest parametru va avea mereu un "/" la final. (Fișierele care vor fi adăugate în arhivă sunt listate în fișierul **files.txt**).

Dacă arhiva a fost creată cu succes, se va afișa la terminal textul "> *Done!*", altfel se va afișa "> *Failed!*".

O arhivă tar grupează fișierele componente într-o înșiruire de octeți, care rețin o secvență de obiecte asociate fișierelor. Fiecare obiect este compus din 2 părți: header și date. Header-ul este o structură care reține informații despre fișierul asociat, și precedă partea de date, care reprezintă efectiv octeții fișierului. Structura header-ului este prezentată mai jos:

```
union record {
        char charptr[512];
        struct header {
                char name[100];
                char mode[8];
                char uid[8];
                char gid[8];
                char size[12];
                char mtime[12];
                char chksum[8];
                char typeflag;
                char linkname[100];
                char magic[8];
                char uname[32];
                char gname[32];
                char devmajor[8];
                char devminor[8];
        } header;
};
```

Semnificația câmpurilor structurii header (mai multe informații găsiți în documentație):

- name = numele fișierului.
- mode = permisiunile asupra fișierului (traduse în biți conform standard-ului). În cadrul temei, biții *Set UID*, *Set GID* și *Save Text* vor fi mereu 0).

- uid = ID-ul utilizatorului owner.
- gid = ID-ul grupului utilizatorului owner.
- size = dimensiunea fișierului ca număr de octeți.
- mtime = data ultimei modificări (se reține un număr întreg care reprezintă numărul de secunde care au trecut de la 1 Ianuarie 1970 00:00, până la data ultimei modificări).
- chksum = suma între toți octeții ce alcătuiesc header-ul.
- typeflag = tipul fișierului (în cadrul temei va fi mereu 0).
- linkname = numele fișierului.
- magic = conține string-ul "*GNUtar* " (atenție, are un spațiu la final).
- uname = numele utilizatorului owner.
- gname = numele grupului utilizatorului owner.
- devmajor, devminor = caracteristice fișierelor speciale (în cadrul temei vor fi mereu 0).

Informațiile ce trebuie stocate în câmpurile header-ului vor fi extrase din fișierele auxiliare **files.txt** și **usermap.txt**, descrise mai jos. Aceste fișiere vor fi în același director cu executabilul **archiver** (ele vor fi adăugate în același director de către checker la rulare). Toate valorile vor fi scrise în octal.

Conform standard-ului, dimensiunea unei arhive tar este multiplu de 512 și dimensiunea fiecărui obiect din cadrul arhivei este multiplu de 512. Așa cum se poate observa, fiecare header al unui obiect are 512 octeți, deci implicit dimensiunea datelor unui fișier trebuie să fie multiplu de 512. În cazul în care un fișier pe care îl arhivăm nu are dimensiunea multiplu de 512, în momentul arhivării, datele acestuia vor fi completate cu 0-uri, până se atinge o dimensiune divizibilă cu 512. Finalul arhivei tar este marcat de o zonă de 512 octeți de 0, pusă după obiectele asociate fișierelor arhivate.

Fișiere auxiliare

- **files.txt**

Fișierul **files.txt** conține fișierele care trebuie puse în arhivă. El conține un subset de linii din rezultatul aplicării comenzii `"ls -la --time-style=full-iso | grep ^-"` în directorul care conține fișierele, astfel că structura acestui fișier este de forma:

```
-rw-rw-r-- 1 user user     3 2019-11-12 17:55:15.882089310 +0200 image1.png
-rw-rw-r-- 1 user user    73 2019-11-12 17:56:10.190055687 +0200 image2.png
-rw-rw-r-- 1 user user   328 2019-11-12 17:57:23.435434132 +0200 image3.png
```

Fiecare linie conține informații despre un singur fișier și are următoarea structură:

```
permissions no_links owner_name owner_group size last_change_time name
```

Mai multe detalii despre formatul întors de `"ls -l"` găsiți aici. În cadrul temei se presupune că toate fișierele care urmează să fie arhivate au același owner_name (utilizator) și același owner_group, și că toate fișierele au drept de citire pentru toți utilizatorii. De asemenea se va face presupunerea că toate fișierele incluse în **files.txt** există.

- **usermap.txt**

Fișierul **usermap.txt** are informații despre utilizatorii sistemului, și mai precis, conține rezultatul comenzii "`cat /etc/passwd`". Fișierul /etc/passwd este un fișier text, care conține o listă a conturilor tuturor utilizatorilor, oferind pentru fiecare cont câteva informații utile precum ID-ul utilizatorului, ID-ul grupului, etc. Structura fișierului arată conform exemplului de mai jos.

```
[...]
sshd:x:114:65534::/var/run/sshd:/usr/sbin/nologin
[...]
admin:x:1003:118::/home/admin:/bin/sh
[...]
mysql:x:116:127:MySQL Server,,,:/nonexistent:/bin/false
[...]
ntp:x:121:135::/home/ntp:/bin/false
[...]
user_name:x:1008:1008:User Name,,,:/home/user_name:/bin/bash
```

Fiecare linie conține informații despre un singur cont și are următoarea structură:

`username:encrypted_password:uid:gid:full_name:home_directory:shell`

Mai multe informații despre semnificația fiecărui câmp găsiți aici. Acest exemplu nu conține fișierul integral, `[...]` reprezintă faptul că acolo mai există niște linii care au fost omise, pentru că nu erau relevante în înțelegerea conceptului.

Cerința 2 - list

Adăugați programului scris anterior funcționalitatea asociată comenzii **list**. Sintaxa acestei comenzi este:

```
list nume_arhiva
```

Această comandă va afișa la standard output numele tuturor fișierelor care sunt incluse în arhiva cu numele *nume_ arhivă*, în ordinea în care sunt salvate în fișierul tar. Se va afișa câte un nume de fișier pe linie, precedat de textul "> ". Dacă arhiva nu există se va afișa textul "> *File not found!*". În urma acestei comenzi, conținutul arhivei rămâne nemodificat.

Pentru a obține rezultatul pentru această comandă se vor parcurge octeții din arhivă, nu liniile din fișierul *files.txt*. În cazul în care această regulă nu este respectată, punctajul obținut pe cerința 2 va fi anulat.

Un posibil scenariu de execuție al temei este în exemplul de mai jos:

```
user@user-pc:~\$ ./archiver
create arhiva1.tar imagini_marte/
> Done!
list arhiva1.tar
> marte1.png
> marte2.png
> marte3.png
create arhiva2.tar imagini_terra/
> Done!
list arhiva2.tar
> terra1.jpg
> terra2.jpg
> terra7.jpg
> terra8.jpg
> terra9.jpg
exit
user@user-pc:~\$
```

Cerința 3 - extract

Adăugați programului scris anterior funcționalitatea asociată comenzii **extract**. Sintaxa acestei comenzi este:

```
extract nume_fisier nume_arhiva
```

Această comandă va extrage conținutul fișierului *nume_fisier*, din arhiva cu numele *nume_arhiva*. Comanda va crea un nou fișier cu numele ***extracted_****nume_fisier* în același director cu executabilul *archiver*. Fișierul va fi scris în format binar, așa cum este salvat în arhiva *nume_arhiva* (headerul nu va fi inclus). Dacă fișierul a fost extras cu succes se va afișa la terminal textul *"> File extracted!"*. Dacă arhiva nu există sau fișierul nu face parte din arhivă se va afișa textul "*> File not found!*". În urma acestei comenzi, conținutul arhivei rămâne nemodificat.

Un posibil scenariu de execuție al temei este în exemplul de mai jos:

```
user@user-pc:~\$ ./archiver
create arhiva1.tar fisiere_text/
> Done!
extract f0.txt arhiva1.tar
> File not found!
extract f1.txt arhiva1.tar
> File extracted!
exit
user@user-pc:~\$
```

Regulament

Regulamentul general al temelor de la Programarea Calculatoarelor se găsește pe ocw, secțiunea Teme de casă. Vă rugăm să îl citiți integral înainte de continua cu regulile specifice acestei teme.

Arhivă

Soluția temei se va trimite ca o arhivă **zip** cu numele în formatul: **Grupa_NumePrenume_TemaX.zip** (Exemplu: `317CA_NumePrenume_Tema3.zip`).

Arhiva trebuie să conţină în directorul rădăcină doar următoarele:

- Codul sursă al programului vostru (fișierele **.c** și eventual **.h**).
- Un fișier **Makefile** care să conțină regulile **build** și **clean**.
  - Regula **build** va compila codul și va genera executabilul **archiver**.
  - Regula **clean** va șterge **toate** fișierele generate la compilare (executabile, binare intermediare etc).
- Un fișier **README** care să conțină explicația implementării alese de voi. Acesta trebuie să prezinte modul în care ați rezolvat cerințele, nu bucăți din enunț.

Arhiva temei nu va conține: fișiere binare, fișiere de intrare/ieșire folosite de checker, checkerul, orice alt fișier care nu este cerut mai sus.

Numele şi extensiile fişierelor trimise nu trebuie să conţină spaţii sau majuscule, cu exceptia fișierului README (care este are numele scris cu majuscule și nu are extensie).

Nerespectarea oricărei reguli din secțiunea **arhivă** implică penalizări asupra punctajului.

Checker

Pentru corectarea aceste teme vom folosi scriptul **check** din arhiva **check_archiver.zip** din secțiunea de resurse asociată temei. Pentru a rula checkerul folosiți comanda `./check`. Acesta folosește **python2.7** și utilitarul **valgrind** care trebuie instalate înainte de a rula checkerul.

Punctaj

Această temă este notată cu maximum **120p**. Pentru obținerea notei 10, este necesar să se obțină 100p/120p, restul se consideră bonus, astfel încât nota maximă posibilă este 12.

Distribuirea punctajului:

- Cerința 1: 40p
- Cerința 2: 20p
- Cerința 3: 20p
- Eliberarea memoriei: 20p (10p cerința 1, 5p cerința 2, 5p cerința 3)
- Claritatea explicaţiilor și a codului: 20p

Modalitatea de acordare a punctajului:

- Punctajul pe teste este cel acordat de **check**, rulat pe vmchecker. Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare).

- Punctajul pe calitatea explicațiilor și a codului se acordă în mai multe etape:
  - **corectare automată**
    - Checkerul va încerca să detecteze în mod automat problemele legate de coding style și de organizare a codului. Acesta va acorda maxim 20p dacă nu sunt detectate probleme.
    - Checkerul va penaliza de asemenea warninguri la compilare sau alte probleme descoperite la runtime.
  - **corectare manuală**
    - Tema va fi corectată manual pentru a verifica aspectele pe care checkerul nu le poate testa. Recomandăm să parcurgeți cu atenție tutorialul de coding style de pe ocw.
    - Codul sursă trebuie să fie însoțit de un fișier README care trebuie să conțină informațiile utile pentru înțelegerea funcționalității, modului de implementare și utilizare a programului. Acesta evaluează, de asemenea, abilitatea voastră de a documenta complet și concis programele pe care le scrieți. Evaluarea fișierului README va fi făcută de către echipa de asistenți, care, în funcție de calitatea documentației, pot aplica depunctări sau bonusuri.
    - Deprinderea de a scrie cod sursă de calitate este un obiectiv important al acestei materii. Sursele greu de înțeles, modularizate neadecvat sau care prezintă hardcodări care pot afecta semnificativ mentenabilitatea programului cerut, vor fi depunctate adițional.
    - În cadrul etapei de corectare manuală se pot aplica depunctări mai mari de 20p.

Trimitere

- Tema **nu** are deadline **soft**, ci doar deadline **hard** pe 13.12.2019 ora 23:55. Nu se mai acceptă nicio trimitere după această dată. (nici măcar cu depunctări).
- Arhiva temei trebuie trimisă pe site-ul de cursuri acs.curs.pub.ro și pe vmchecker.
- Tema poate fi submisă de oricâte ori fără depunctări până la deadline. Mai multe detalii se găsesc în regulamentul de pe ocw.
- O temă care nu compilează pe vmchecker nu va fi punctată.
- O temă care compilează, dar care nu trece niciun test pe vmchecker, nu va fi punctată.
- Punctajul pe teste este cel acordat de scriptul **check** rulat pe vmchecker. Așa cum este menționat și mai sus, echipa de corectare își rezervă dreptul de a depuncta orice încercare de a trece testele în mod fraudulos.
- Ultima temă trimisă pe vmchecker poate fi rulată de către echipa de corectare de mai multe ori pentru a verifica corectitudinea și robustețea soluției. Vă recomandăm să rulați **local** tema de mai multe ori pentru a verifica faptul că punctajul obținut este mereu același.