Programarea calculatoarelor (2019-2020)
Tema 1 - Pachet de utilitare din linia de comandă

Deadline: 09.12.2019, ora 23:55

Responsabili temă: Simescu Andrei, Vlad Botezatu, Mihai Nan

Adrian este un vestit inginer în domeniul Calculatoarelor, motiv pentru care Marcel, un tânăr student, vrea sa îi fie ucenic. Știind că Adrian lucrează adesea cu calculatoare care rulează pe platforme variate, Marcel și-a propus să îi implementeze un pachet de utilitare de bază UNIX scrise în C, pentru a-l convinge pe Adrian să îl ia drept ucenic. Din păcate, momentan, Marcel nu excelează la domeniul programare, așa că v-a cerut vouă ajutorul.

Cerința 1

Pentru început, Marcel vrea să realizeze funcția **grep**, pentru a căuta mai ușor informații. Aceasta primește la tastatură un șir de caractere căutat, apoi numărul de linii al textului și, în final, textul în care se va căuta, linie cu linie. Programul afișează fiecare linie care conține cel puțin o apariție a șirului căutat, colorând aparițiile subșirului cu roșu, exact precum o face și utilitarul linux "**grep**".

Datele de intrare citite de la tastatura vor fi salvate corespunzător, astfel:

-Șirul de căutat are dimensiune maximă 30 și va fi salvat într-un vector de dimensiune fixă, care nu este necesar să fie alocat dinamic, decât dacă voi doriți acest lucru. Acest șir poate conține spații.

- Numărul de linii va fi salvat într-o variabila n, cu proprietatea că 0<n<100.

- Liniile citite în continuare vor fi salvate într-un vector de pointeri la șiruri, dimensiunea maximă a vectorului fiind n, iar pentru fiecare linie în parte se va aloca exact atata memorie cat este necesară.

Se considera ca o linie nu va avea mai mult de 200 de caractere.

Observație: În momentul în care găsiți un număr x de apariții ale șirului căutat în unul din elementele vectorului (adică pe una din linii), va trebui să realocați memoria, mărind dimensiunea acestuia cu x * numărul de caractere introduse pentru colorarea șirului, pentru a putea face astfel modificările necesare elementelor din vector fără să ieșiți din memorie. Aceste rânduri au dimensiune maximă de 200 de caractere (cu tot cu modificările aduse).

HINT:

- Dacă doriți să nu introduceți de fiecare data aceleași informații în linia de comandă, recomandarea noastră este să folosiți redirectarea de fișiere de intrare la stdin, după cum ați învățat la USO.
- Tutorial despre cum se adaugă culorile în **printf** și alte situații în sistemele UNIX: http://www.andrewnoske.com/wiki/Bash_-_adding_color

Pentru ca tema sa poată fi testată corespunzător, vom folosi culoarea roșie implicit, adică textul de colorat va fi încadrat între "\e[0;31m" și "\e[m" (pentru resetare).

ATENȚIE:

'\e' este un caracter special (nu două), mai precis, caracterul ESCAPE (cod ascii 27). Din această cauză, dacă încercați să redirectați outputul într-un fișier, acest caracter fie nu va fi vizibil, fie va fi reprezentat printr-un pătrat/caracter indescifrabil (în funcție de editorul de text ales și de setările folosite). Astfel, atunci când faceți redimensionarea șirului de caractere în care colorați, veți adaugă 10 caractere (7 în față, respectiv 3 în spate), și nu 12, cum am fi tentați să credem.

Restricții și precizări:

`strlen(lookup_string) < 30`

`n < 100`

Rândurile vor avea maxim 200 caractere(cu tot cu cele 10 * numărul de apariţii ale şirului căutat în rândul curent)

Exemplu:

INPUT	***OUTPUT***	***OUTPUT VĂZUT ÎNTR-UN FIȘIER***
Ana 3 Ana are mere Maria are mai multe mere Georgiana e prietena cu Ana	Ana are mere Georgiana e prietena cu Ana	[0;31mAna[m are mere Georgiana e prietena cu [0;31mAna[m

Cerința 2

Mai departe, Marcel dorește să implementeze funcția **cut**, pentru a putea extrage diferite informații din tabele de toate tipurile. Acesta va primi de la tastatură un text ale cărui linii trebuiesc împărțite în mai multe câmpuri, în funcție de un șir de delimitatori, după care va selectata doar câmpurile menționate într-un șir de numere reprezentând indicii câmpurilor respective, numerotate de la 1, iar între aceste câmpuri, Marcel va plasa un alt delimitator, numit delimitator de ieșire.

Datele de intrare ce vor fi citite de la tastatură sunt:

-pe prima linie, șirul de delimitatori, ***del***, care trebuie să conțină ca delimitator și ***'\n'***

-pe a doua linie, dimensiunea șirului de indici ai câmpurilor ce trebuie selectate, ***n***

-pe a treia linie, șirul de indici, ***fields***

-pe a 4-a linie, șirul reprezentând delimitatorul de ieșire, ***out_del***, care nu va conține și caracterul '\n

-pe a 5-a linie, numărul de linii, ***m***, ale textului

-următoarele **m** linii vor forma textul pe care se va aplica funcția **cut**, care vor fi salvate într-un vector de pointeri la șiruri de caractere, cu număr fix de rânduri (m), dar fiecare rând va avea lungime variabilă.

Șirul în care sunt specificați indicii campurilor selectate este un șir de numere delimitate prin virgulă, despre care nu se garantează nici ca sunt în ordine crescătoare și nici că definesc un câmp care există, după cum se vede și în exemplu (câmpurile 6 și 7 există doar pentru ultimele două rânduri, în timp ce câmpul 20 nu există pentru niciuna din liniile textului).

Deși **cut** din linux poate să identifice și câmpuri goale (doi delimitatori unul după altul), pentru a putea folosi mai ușor funcțiile de baza din biblioteca *string*, se garantează că nu vor exista astfel de situații.

Restricții și precizări:

`strlen(del) < 10`

`n < 100`

`strlen(fields) < 200`

`strlen(out_del) < 10`

`m < 100`

Rândurile vor avea maxim 200 caractere

Exemplu:

INPUT	***OUTPUT***
: 6 5,1,3,6,7,20 -- 4 Ana:18:Verde:Mere:Bucuresti Marcel:20:Albastru:Banane:Buzau Alex:31:Alb:Struguri:Londra:Anglia:Septembrie Adrian:67:Maro:Kiwi:Pitesti:Romania:August	Ana--Verde--Bucuresti Marcel--Albastru--Buzau Alex--Alb--Londra--Anglia--Septembrie Adrian--Maro--Pitesti--Romania--August

Cerința 3

Mulțumit fiind de munca sa de până acum, Marcel hotărăște să ia o pauză de la pachetul de utilitare și să își facă un program pentru afișarea mediilor sale din liceu. Deoarece este o fire optimistă, vrea să îi fie afișate mai întâi notele unde are mediile cele mai mari.

Deja are o matrice a notelor construită astfel incat pe o linie se afla notele corespunzătoare unei materii, așa ca voi va trebui doar să o citiți, să sortați liniile strict crescător în funcție de medie și să le afișați (dacă la două materii Marcel are aceeași medie, se va păstra ordinea în care au fost citite), afișând și media la începutul fiecărei linii, cu 3 zecimale exacte, pe exact 10 spații, aliniată la stânga, nu la dreapta după cum este predefinit.

Materiile din liceu au un număr diferit de note, astfel că fiecare set de note de pe un rând se va încheia cu numărul 0.

Matricea de note va fi complet alocată dinamic, exact atata memorie cata este necesara atat pentru notele fiecarei materii cat și ca număr de linii (materii). Numărul de materii va fi introdus de la tastatură pe prima linie, urmat de matricea notelor.

Restricții:

n < 100

Un rând va conţine maxim 200 de caractere (ATENŢIE: nu 200 de numere!)

NOTĂ: Lungimea unei linii din matrice nu vă este dată, deci va trebui să alocați și să realocați dinamic această mărime pentru fiecare linie, folosind o dimensiune maximă inițială 3, pe care apoi să o incrementati cu 3 de fiecare dată când numărul de elemente depășește dimensiunea deja alocată.

Exemplu:

INPUT	***OUTPUT***
4 8 7 10 8 0 10 6 0 10 10 10 10 10 10 0 9 9 10 9 10 0	10.000 10 10 10 10 10 10 9.400 9 9 10 9 10 8.250 8 7 10 8 8.000 10 6

Cerința 4

În final, pentru a-și proteja importantele proiecte inginerești de persoane care ar vrea să i le fure, Marcel s-a gândit să implementeze pentru Adrian un simplu program de codificare a textelor.
Algoritmul la care s-a gândit este următorul: numerele (șiruri de caractere separate prin spațiu sau "\n" care conțin doar cifre) care apar în text, vor rămâne neschimbate, iar pentru fiecare cuvânt (orice șir separat prin spațiu sau "\n", care nu este număr) trebuie să păstrați prima literă pe loc, deoarece aceasta va juca rolul de cheie: toate celelalte caractere din cuvânt vor fi adunate cu cheia și se va realiza mod 256 pe suma obținută (se garantează că această sumă nu va fi niciodată multiplu de 256). **NOTĂ: "-12345"** se consideră cuvânt, la fel și "**mere,**".
Se va afișa textul codificat.

Datele se vor citi de la tastatură. Pe prima linie se va citi numărul ***n*** al liniilor din text, urmat de cele ***n*** linii.

Restricții:

```
n < 100
Un rând conţine maxim 200 de caractere
```

Precizări:

Caracterul special '\n' nu va fi codificat, și chiar va trebui reținut, întrucât șirurile codificate vor fi plasate corespunzător cu șirurile de intrare, după cum se observă și în exemplu.

Șirurile vor fi salvate într-un vector de șiruri de caractere, dimensiunea vectorului fiind n fixă, iar pentru fiecare linie în parte se va aloca exact atata memorie cat este necesară.
Se considera ca o linie are maxim 200 de caractere.

Citirea fiecărui rând se va face folosind fgets(sirul_in_care_salvez_randul, 200 , stdin)

Exemplu:

INPUT	***OUTPUT***
3 *.ANA* *!ARE 12345 &MERE* *%6-*58 83210*	*.o|o* *!bsf 12345 &skxk* *%[ROZ] 83210*

Makefile

Veți primi și un makefile, care, la rularea comenzii make, va compila toate codurile sursă, astfel:

-Fișierele ce conțin coduri sursă se numesc grep.c, cut.c, sort.c, encrypt.c.

-După rularea comenzii make, fișierele executabile create vor avea numele grep, cut, sort, encrypt.

-Pentru rularea fiecărui task, se vor folosi comenzile:

./grep -> pentru cerința 1
./cut -> pentru cerința 2
./sort -> pentru cerința 3
./encrypt -> pentru cerința 4

-La rularea comenzii make clean, fișierele executabile vor fi șterse din folderul curent.

Date de intrare

Datele de intrare sunt descrise pentru fiecare cerință individual, și vor fi cititie de la tastatura folosind scanf unde este cazul (cuvinte în care se poate ignora spațiile sau '\n'), în timp ce liniile vor fi citite folosind fgets(șirul_in_car_salvam, 200, stdin).
Pentru a nu fi nevoiți să scrieți de fiecare dată inputul, și nici să nu stați să dați copy paste, puteți redirecta un fișier în care să scrieți datele pe care de obicei le introduceati de la tastatură, și să rulați folosind formatul următor:
./numele_problemei_rezolvate < numele_fisierului_de_intrare

Date de ieșire

Se vor afișa în consolă răspunsurile pentru fiecare cerință, conform specificațiilor din enunț și exemple.
La testare, outputul va fi redirectat într-un fișier pentru a-l compara cu rezultatul oferit în arhiva de teste.
Astfel, dacă întâmpinați probleme la checker deși outputul vostru pare bun, recomandăm redirectarea într-un fișier și compararea acelui output cu rezultatul corect, astfel:
./numele_problemei_rezolvate < numele_fisierului_de_intrare > numele_fisierului_de_ieșire

Indicații de rezolvare

- Pentru fiecare din cele 4 cerințe există un exercițiu în laborator de la care să porniți sau care să vă ofere o oarecare idee de rezolvare.
- Cele 4 cerințe sunt independente una de cealaltă, așa că, dacă întâmpinați probleme la vreuna dintre ele, puteți să începeți rezolvarea altei cerințe.
- Pentru primele două cerințe puteți să studiați modul de utilizare ale funcțiilor **grep** și **cut** din sistemele **UNIX**, întrucât voi veți implementa ceva asemanator, cu toate acestea, nu trebuie să vă abateți de la detaliile oferite în cadrul cerinței, deoarece există mici diferențe menite să vă facă rezolvarea mai ușoară.
- Pentru sortare, puteți folosi atât **qsort,** cât și orice altă funcție de sortare învățată de voi. Pentru ușurință și pentru faptul că v-ați obișnuit cu ea la laborator, va recomandăm totuși utilizarea funcției **bubble sort.**

Restricții și precizări

- **Separați logica programelor în mai multe funcții! Veti avea functii pentru citire date intrare, prelucrari si afisare a rezultatelor, eliberarea memoriei, interschimbari etc. Nu se vor puncta sursele in care tot programul este scris in main!**
- **Este interzisă folosirea variabilelor globale.**
- Soluția temei va fi scrisă în C. Nu folosiți sintaxă sau instrucțiuni specifice limbajului C++.
- Rezolvarea temei va fi scrisă în fișierele aferente fiecărui exercițiu.

- Temele sunt strict individuale. Copierea temelor va fi sancționată. Persoanele cu porțiuni de cod identice, nu vor primi niciun punctaj pe temă.
- Pentru întrebări despre temă se va folosi în mod exclusiv forumul temei, pe care vă recomandăm să îl vizitați chiar și dacă nu aveți întrebări, întrucât este posibil să aflați informații noi din întrebările puse de colegii voștri, respectiv din răspunsurile date de noi.
- Temele trimise după deadline nu vor fi luate în considerare.

- În ***README*** precizați cât timp v-a luat implementarea programului vostru și explicați, pe scurt, implementarea temei (comentariile din cod vor documenta mai amănunțit rezolvarea).
- Este recomandat ca liniile de cod și cele din fișierul README să nu depășească 80 de caractere.
- Folosiți un coding style astfel încât codul să fie ușor de citit și înțeles. De exemplu:
    - Dați nume corespunzătoare variabilelor și funcțiilor.
    - Nu adăugați prea multe linii libere sau alte spații goale unde nu este necesar (exemplu: nu terminați liniile în spații libere, trailing whitespaces; nu adăugați prea multe linii libere între instrucțiuni sau la sfârșitul fișierului). Principalul scop al spațiilor este indentarea.
    - Fiți consecvenți. Coding style-ul are o natură subiectivă. Chiar dacă pentru unele persoane nu pare bun, faptul că îl folosiți consecvent este un lucru bun.
    - Există programe sau extensii pentru editoare text care vă pot formata codul. Deși vă pot ajuta destul de mult, ar fi ideal să încercați să respectați coding style-ul pe măsură ce scrieți codul.
    - – Pentru sugestii de coding style, puteți intra aici și aici.

- Veţi trimite o arhivă ZIP cu numele de tipul ***GRUPA_Nume_Prenume.zip*** (exemplu: ***311CC_Popescu_Maria_Ioana.zip***), care va conţine fișierele necesare, ***Makefile*** și ***README***. Fișierele trebuie să fie în rădăcina arhivei, nu în alte subdirectoare!!!
- Compilarea nu ar trebui să producă avertizări (verificați prin adăugarea flagului ***-Wall*** la gcc).

- **Eliberați memoria alocată dinamic**. Folosiți comanda de mai jos pentru a verifica dacă memoria este eliberată corect.
  `valgrind --tool=memcheck --leak-check=full ./nume_cerinta < fisier_input`
- **Temele trebuie să fie încărcate pe vmchecker. NU se acceptă teme trimise pe e-mail sau altfel decât prin intermediul vmchecker-ului.**

Notare:

- **25p** - Cerința 1
- **40p** - Cerința 2
- **25p** - Cerința 3
- **35p** - Cerința 4
- **25p** - *Coding style*, *Valgrind*, *README*
    - **10p** - **Coding Style**
        - Comentarii
        - Folosirea a mai multor funcții
        - Logică clară
        - Spațiere corectă
        - Stil **consecvent**
        - Variabile și funcții denumite adecvat
    - **10p** - Eliberarea și accesarea corectă a memoriei
    - **5p** - Completarea fișierului **README**