Fitxategi-sistema eta Sarrera/Irteera - Ariketak

1. ***kopiatu*** programa aldatu behar da, zero, bat edo bi argumentu onar ditzan. Argumentu bakarra pasatzen bazaio, irteera estandarrean erakutsi behar den fitxategiaren izena izango da. Bi argumentu pasatzen bazaio, sarrera eta irteera fitxategien izenak izango dira, sarrerako fitxategiaren edukia irteerako fitxategian kopiatu beharko delarik. Argumenturik ez bazaio pasatzen, adibideetako ***kopiatu*** bezala exekutatuko da.

```
kopiatu [sarrera_fitxategia [irteera_fitxategia]]
```

2. ***ezabatu*** programa idatz ezazu (Unix-eko ***rm*** antzekoa), argumentu bezala pasatako fitxategiak ezabatzen dituena. Gutxienez argumentu bat izan behar du.

```
ezabatu fitx1 [fitx2 ... [fitxn]]
```

3. Alda ezazu buztana programa, erakutsitako karaktere kopurua argumentua (ez derrigorrezkoa) izan dadin.

```
buztana [-n] fitxategia
```

4. ***filtroa*** komandoa programatu behar da (Unix-eko ***grep*** antzekoa), sarrera estandarretik lerroak irakurtzen duena eta irteera estandarrean soilik pasatako patroia dutenak idazten duena. (Oharra. Lerroak irakurtzen duen funtzioa erabiltzea komeni da, sarrera estandarra fitxategi batera berbideratu ezkero ongi funtzionatu dezan).

```
filtroa patroia
```

5. ***berdinak*** komandoa programatu behar da, bi fitxategi konparatzen dituena, 0 bueltatuz berdinak badira, eta -1 berdinak ez badira.

```
berdinak fitx1 fitx2
```

6. ***paperontzia*** programa inplementatu behar da, sarrera estandarretik fitxategien izenak irakurtzen dituena, eta fitxategi bakoitza `/etc/paperontzia` katalogora mugitzen duena (bertan izen berdineko fitxategia badago, zaharra galdu egiten da eta berria gelditzen da).

```
paperontzia
```

7. ***bilatu*** komandoa inplementatu behar da, uneko katalogoan eta lehen mailako azpikatalogoetan, argumentu bezala pasatako izena duten fitxategi arrunt guztiak irteera estandarrean idazten duena (izen osoa, bidea barne).

```
bilatu fitxategia
```

8. ***katalogoa_ezabatu*** komandoa inplementa ezazu, uneko katalogoan argumentu bezala pasatako katalogoa bilatzen duena. Aurkitzen badu, utzik dagoela egiaztatzen du, eta horrela bada ezabatzen du. Katalogo bat utzik dagoen egiaztatzea ezin da fitxategiaren tamainari begiratuz egin, baizik eta bere barruko elementuak kontatuz (barruan elementurik ez duela egiaztatu behar da).

```
katalogoa_ezabatu katalogoa
```

9. ***exekutagarria_da*** funtzioa programatu behar da, pasatako fitxategia arrunta bada eta exekuzio baimena badu 0 bueltatzen duena, eta 1 bestela.

```
int exekutagarria_da(char *izena);
```

10. ***exekutagarrien_zerrenda*** programa inplementatu behar da, irteera estandarrean argumentu bezala pasatako katalogoaren fitxategi exekutagarrien izenak idazten dituena. Aurreko ariketako ***exekutagarria_da*** funtzioa erabil ezazu.

```
exekutagarrien_zerrenda katalogoa
```

11. Unix-eko oinarrizko *shell*-a hobetzea pentsatu dugu. Oinarrizko *shell*-ean fitxategi edo katalogo bat izenez aldatu edota ezabatu nahi dugunean, fitxategi edo katalogoaren izen osoa idatzi behar dugu. Hau erraztu nahi da hurrengo moduan: erabiltzaileak fitxategi edo katalogoaren izena idazten hasten da, eta ESCAPE tekla sakatzean *shell*-ak izena osatzen du, hasiera berdina duten fitxategien artean egon daitezkeen anbiguotasunak kontuan harturik. Adibidez, uneko katalogoan `testu.txt`, `azterketa.txt`, eta `testu1.txt` fitxategiak baditugu, honako emaitzak izan beharko genituzke:

`rm az[ESC]`	`rm azterketa.txt` exekutatzea bezala
`rm testu1[ESC]`	`rm testu1.txt` exekutatzea bezala
`rm tes[ESC]`	Anbiguotasun arazoa `testu.txt` eta `testu1.txt` artean (`rm tes` exekutatzea bezala)

Hobetutako *shell*-aren parte izango den honako funtzioa programatzea eskatzen da:

```
int izena_osatu(char *izen_zatia, char *izen_osoa);
```

Funtzioak parametro bezala pasatako izen zatia osatzen saiatzen da, azaldu berri den moduan. Pasatako izen zatiarekin hasten den fitxategirik ez bada aurkitzen, orduan 1 bueltatuko du. Izena osatzerakoan ez bada anbiguotasunik aurkitzen, 0 bueltatuko du (`izen_osoa` behar den balioa duelarik, noski). Anbiguotasuna aurkitzen bada, 2 bueltatuko du (kasu honetan `izen_osoa` parametroan `izen_zatia` parametroaren edukia bueltatuko da).

12. Unix-eko komando berri bat inplementatu nahi da, ***katm*** izenekoa, irteera estandarrean uneko katalogoko fitxategietatik soilik bere tamaina (bytetan) argumentu bezala pasatako tamaina baino handiagoak direnak idazten dituena.

```
katm tamaina
```

Gogoratu C programetako *main* funtzioari pasatako argumentuak karaktere-kateak direla. Ariketa honen kasuan, karaktere-kate honek zenbaki bat adierazten du.

13. Honako funtzioa inplementatzea eskatzen da:

```
long tamaina(char *izena);
```

Funtzioak argumentu bezala pasatako fitxategiaren tamaina (bytetan) bueltatuko du. Honako deiak erabil itzazu: ***open***, ***lseek*** eta ***close***.

14. ***ukitu*** programa idaztea eskatzen da, fitxategi baten data aldatzen duena uneko data esleituz (Unix-eko ***touch*** komandoa bezala).

```
ukitu fitxategia
```

15. Ondoko formatua duen ***more2*** izeneko komandoa programatu behar da:

```
more2 fitxategia
```

Komando hau exekutatzean UNIX-eko ***more***-aren antzeko emaitza lortu behar da, hau da, fitxategiaren datuak lerroz lerro idazten dira irteera estandarrean pantaila osoa osatu arte (24 lerro). Ondoren itxoiten da erabiltzailea karaktere bat sakatu arte sarrera estandarretik. Sakatutako karakterea '*q*' (*quit*=utzi) baldin bada programaren exekuzioa eten egin behar da, 'ENTER' bada lerro bakar bat gehiago idatziko da, eta gainontzeko kasuetan beste orri oso bat idatziko da. Programaren bukaera-kodea 0 izango da bukaera arrunta denean, 1 '*q*' bidez bukatzen denean, eta 2 erroreren bat gertatzen bada.

Komenta ezazu izandako arazoak ***read*** sistema-deia erabili ezkero sakatutako tekla detektatzeko. Zein da arrazoia?

16. Informazio sistema baten atzipenaren kontrolerako programa bat inplementatu nahi da, ***egiaztatu_pass*** izenekoa. Programa honek sarrera estandarretik erabiltzaile baten identifikadorea (zenbaki osoa) eta gako edo *password*-a (testu-katea) irakurriko ditu, eta atzipena zuzena den ala ez egiaztatuko du, 0 bueltatuz atzipena zuzena bada, eta -1 zuzena ez bada. Sistemako `/usr/net/atzipena` fitxategian konektatu daitezkeen erabiltzaile guztien atzipen informazioak gordetzen dira. Fitxategi honen erregistroen formatua honakoa da:

```
struct s_erabiltzaile {
   int id;
   char password[64];   /* Zifratuta */
}
```

Fitxategi honetan *password*-a zifratuta dago. Zuzena den ala ez egiaztatzeko, jadanik kodetuta dagoen ***zifratu*** funtzioa erabili behar da, parametro bezala pasatako testua zifratuta bueltatzen duena:

```
char *zifratu(char *testua);
```

Fitxategiko erregistroak ordenatuta daude. Gainera, posible den erabiltzaile-identifikadore bakoitzeko erregistro bat existitzen da fitxategian. (Eraginkortasun arrazoiengatik, ezinezkoa da atzipen sekuentziala erabiltzea).

egiaztatu_pass programa inplementa ezazu.

17. UNIXeko Sarrera/Irteerako sistema-deiak eta katalogoak atzitzeko liburutegi errutinak erabiliz, ondoko programa idatzi nahi dugu C lengoaian:

`ls_pertsonalizatua` `katalogo1 katalogo2 … katalogo_N`

Programa honek irteera estandarretik argumentu bezala pasatako katalogoen sarrera guztiak aurkeztuko ditu. Sarrera bakoitzeko honako informazioak idatziko ditu, lerro bakar batean:

```
katalogo_izena/sarrera_izena:tamaina_bytetan:inodo_zenbakia
```

Gerta daitezkeen erroreen kontrola egitea ere eskatzen da.

18. UNIX-eko Sarrera/Irteerako sistema-deiak erabiliz, ondoko programa idazti nahi dugu C lengoaia erabiliz:

`lortu_gakoa erabiltzailea`

Programa honek `erabiltzailea`–ri dagokion informazio-lerroa bilatuko du `ERABILTZAILEAK` izena duen fitxategi batean. Fitxategi honetan lerro bat erabiltzen da erabiltzaile bakoitzaren informazio guztia gordetzeko, bertako edukiaren egitura ondokoa delarik:

```
erab_1  gako_1_posizioa  gako_1_luzera
erab_2  gako_2_posizioa  gako_2_luzera
...
erab_n  gako_n_posizioa  gako_n_luzera
```

Lerro bakoitzeko informazioa zuriunez bananduta dago, eta lerro guztien luzera finkoa dela kontsidera daiteke (40 karaktere lerro bakoitzeko). Lerroaren azken bi eremuak dira, aldez aurre, gako pertsonalaren kokapena eta luzera bytetan. Erabiltzaile guztien gakoak `GAKO_FITX` izeneko fitxategian daudelarik. Badaukagu dagoenekoz kodetuta `lerroa_aztertu` funtzioa, `ERABILTZAILEAK` fitxategiko lerro bat pasatzen badiogu bere hiru informazioak bueltatzen dizkiguna. Bere egitura honakoa da (erabiltzailearen izena 10 karaktereko luzera du):

```
void aztertu_lerroa(char *lerroa, char *erab, long *pos, int *luz);
```

Programak `erabiltzailea`ri dagokion lerroa aurkitzen duenean, `GAKO_FITX` fitxategia atzituko da beste bi informazioak erabiliz, eta erabiltzaileari dagokion gakoa irakurriko du. Azkenik, gakoa irteera estandarrean idatziko da eta 0 balioa itzuliz amaituko da. Erabiltzailea ez bada aurkitzen `ERABILTZAILEAK` fitxategian, mezu bat idatziko da irteera estandarrean eta 1 balioa itzuliz amaituko da. Bestalde, edozein errore egoera topatuz gero (arazoak bi fitxategiak atzitu behar direnean...) mezu bat idatziko da irteera estandarrean eta 2 balioa itzuliz amaituko da.