Beadandó feladat

Due 2023. jan 8 by 23:59

Pont 30 **Beküldés...** egy fájlfeltöltés **Fájltípusok** zip

Elérhető 2022. dec 27, 06:00 - 2023. jan 8, 23:59 13 nap

Ez a feladat zárolva lett ekkor: 2023. jan 8, 23:59 .

A *reverse* parancs megvalósítása

Ebben a feladatban megvalósítjuk a *reverse* parancsot: az argumentumként kapott fájlok tartalmát, vagy a konzolról beolvasott sorokat sorszámozva vagy sorszámozás nélkül, fordított sorrendben, és a sorokat megfordítva (tükrözve) írjuk ki a standard outputra. Pl. ha a *test.txt* tartalma a következő, és sorszámozást is kérünk:

```
alma
barack
szilva
```

…akkor a reverse parancs eredménye a következő:

```
3 avlizs
2 kcarab
1 amla
```

Példa több fájl feldolgozására: ha a *test.txt* fájlt kétszer adjuk meg paraméterül, és nem kérünk sorszámozást, akkor a kimenet a következő:

```
avlizs
kcarab
amla
avlizs
kcarab
amla
```

A megoldáshoz feltesszük, hogy bármely fájl teljes egészében elfér a memóriában.

A megoldás forráskódját (a .c és .h fájlokat, tehát a lefordított binárist **nem** kell) egy zip fájlba tömörítve töltsd fel! A megoldást a határidő lejártáig akárhányszor fel lehet tölteni. A beküldött megoldást automata tesztelő fogja lefordítani. A fordítás eredményét komment formájában láthatod majd a Canvasen.

Fontos: a feladatban leírt kimeneteket pontosan a megadott formátumban adjuk meg, különben az automata tesztelő rossz megoldásnak fogja értékelni a programot!

(*Megjegyzés*: a feladat alapja a *rev* Unix utility, amely a kapott bemenet(ek) tartalmát eredeti sorrendben, de sorszámozás nélkül és a sorokat megfordítva írja ki.)

Feladat

A program olvassa be az első parancssori argumentumként, hogy szeretnénk-e a sorszámokat is kiírva látni a képernyőre! A paraméter két lehetséges értéke legyen “linenums” és “nolinenums”, más stringet ne fogadjunk el. Második parancssori argumentumként olvassuk be egy tetszőleges sor maximális hosszát (a beolvasott karakterek maximális számát)! Ezt a két adatot tároljuk egy struktúrában.

Amennyiben a felhasználó nem ad meg legalább két parancssori argumentumot, jelenítsük meg a következő használati utasítást (a második sor egy tabulátorral kezdődik), majd fejeződjön be a program:

```
Usage:
    rev [SHOW LINE NUMBERS] [MAX LINE LENGTH] files...
```

A beolvasandó fájlok nevét további parancssori argumentumokként adjuk meg a programnak, és olvassuk be egyenként a fájlok tartalmát. Ha az aktuális argumentum hibát okoz (a fájlt nem sikerült megnyitni), írjunk hibaüzenetet a standard error outputra, és folytassuk a végrehajtást a következő argumentummal. A hibaüzenet legyen: `File opening unsuccessful: <fájl neve>`. Ha a fájl egy sora hosszabb, mint a maximális sorhossz, a fennmaradó karaktereket hagyjuk figyelmen kívül!

Ha a felhasználó nem adott meg egy fájlnevet sem, olvassunk be sorokat a standard inputról, amelyek legfeljebb a megadott maximális számú karakterből állnak! Ebben az esetben ne írjunk a konzolra semmit, csak várjunk a felhasználó inputjára.

(Tipp 1: a fájl típusú pointer valójában egy stream, amely helyettesíthető a standard inputtal (`stdin`).)

(Tipp 2: az EOF esemény Unix rendszeren a Ctrl+D, Windowson a Ctrl+Z parancssal váltható ki.)

A sorok tárolásához használjunk dinamikus tömböt! A tömb létrehozásakor a mérete legyen egy előre rögzített érték (pl. 8). Ha a memóriafoglalás sikertelen, írjunk ki hibaüzenetet (`Memory allocation failed!`), és fejezzük be valamilyen hibakóddal a végrehajtást. (Ugyanez vonatkozik a tömb méretének megváltoztatására.) Ne számoljuk meg előre az aktuális fájl sorait, hanem duplázzuk meg a tömb méretét, amennyiben a sorok száma túllépi az aktuális méretet!

A megoldást bontsd több fordítási egységre! A `main` függvény kerüljön a `main.c` fájlba, minden más függvényimplementáció kerüljön külön fordítási egységbe, amelyhez készüljön el egy header állomány. A header állományt védd header guarddal.

Elvárások a programmal szemben

- A nem forduló kód automatikusan 0 pontot ér. (Természetesen ez csak a legutoljára feltöltött megoldásra vonatkozik, a feladat határideje után.)
- Ne használj globális változókat!
- Logikusan tagold a megoldást. A megoldás részeit külön függvényekben valósítsd meg.
- Ügyelj, hogy ne legyen a programban memóriaszivárgás!
- Kerüld a nem definiált viselkedést okozó utasításokat!

Tanácsok

- Ne feledkezzünk meg a dinamikus memóriafoglalás sikerességének ellenőrzéséről, és a foglalt memória felszabadításáról! Teszteld a programot `valgrind`dal, hogy felderítsd az esetleges memóriaszivárgást.
- Amennyiben a beolvasáshoz az `fgets` függvényt használod, ne feledd, hogy ha a sor rövidebb, mint a puffer maximális hossza, a függvény beolvassa a sortörést is, ami a kiírásnál hibás eredményhez vezethet.