

1. Javítsa ki az összes ismétlés hibát a gy0.zip-ben található open_fopen.c mintaprogramban! A gy0.zip-ben mellékelte a text.txt fájl utolsó sora után nincs soromelés. Ha ezt a fájlt adjuk át a program paramétereként, akkor a „Using fopen - binary” kiíratás beduplázza az utolsó sor utolsó karakterét. Ha a text.txt utolsó sora után beszúrunk egy soromelést, akkor pedig a „Using fopen - reads in lines” kiíratás duplázza be az utolsó sort. E két hibát kell kijavítani úgy, hogy egyik se duplázzon, függetlenül attól, hogy van-e soromelés az utolsó sor végén!
2. Készítsen programot, mely létrehoz egy text fájlt, és beleírja a program készítőjének nevét! Ha a fájl létezik, akkor semmisítse meg az előző tartalmát, és ezt követően írja be a készítő nevét!
3. Készítsen programot, mely a következő tevékenységeket hajtja végre:
 - Definiáljon egy tetszőleges struct-ot két int típusú adattaggal,
 - adjon az adattagoknak értéket,
 - írja ki fájlba,
 - olvassa vissza (írja a visszaolvasott értékeket a konzolra),
 - végül ellenőrizze, hogy azt olvasta ki, amit beírt!
4. Módosítsa az előző feladatban leírt programot úgy, hogy a számokat a program paraméterben kérje be! (Szükség lehet char* -> int konverzióra, használhat atoi-t.)
5. Készítsen programot, mely módosítja egy fájl utolsó karakterét! A fájl nevét paraméterként kapja. Ellenőrizze, hogy a fájl létezik-e, és van-e joga módosítani a tartalmát! Szükség esetén tájékoztassa a felhasználót az esetlegesen felmerült problémáról! Azt a karaktert, amire a fájl utolsó karakterét módosítja, konzol inputként olvassa be! Módosítás előtt jelenítse meg a módosítandó karaktert.

Minta:

```
klettner@szamrend:~$ cat >f5a.txt
Hajrá Fradi!klettner@szamrend:~$ cp f5a.txt f5.txt
klettner@szamrend:~$ chmod 000 f5a.txt
klettner@szamrend:~$ ls -l f5*
----- 1 klettner oktatok 13 febr  26 17:32 f5a.txt
-rw-r--r-- 1 klettner oktatok 13 febr  26 17:32 f5.txt
klettner@szamrend:~$ a.out
Kérem pontosan egy paramétert adjon meg!
Használat: a.out {fájl}
klettner@szamrend:~$ a.out f5
f5 nevű fájl nem létezik!
klettner@szamrend:~$ a.out f5a.txt
Nem rendelkezik a szükséges hozzáférési jogosultsággal!
klettner@szamrend:~$ a.out f5.txt
Lecserélendő karakter: !
Kérem az új karaktert: ?
klettner@szamrend:~$ cat f5.txt
Hajrá Fradi?klettner@szamrend:~$
```

6. Készítsen programot, mely soronként olvassa ki egy forrás fájl tartalmát, majd a sorokat rendezve írja ki egy cél fájlba! A fájlok neveit a konzolról olvassa be futás közben.

Ötletek a megoldáshoz: Olvassa végig a fájl tartalmát a sorvég jeleket figyelve, miközben megszámlálja, hány sorból áll a fájl, és mekkora a leghosszabb sor. Hozzon létre olyan string elemtípusú tömböt, melynek elemszáma megegyezik a fájl sorainak számával, és elemmérete lehetővé teszi a leghosszabb sor tárolását is. Olvassa a tömbbe a fájl sorait, rendezze a tömböt, majd írja ki a rendezett tömb sorait a cél fájlba. Ügyeljen rá, hogy egy szövegfájl utolsó sora nem minden esetben zárul soremeléssel!

7. Készítsen programot, mely egy char és int adattagokból álló struktúra alkotta n (konstans, páros) elemű tömböt rendre feltölt 'a'-tól kezdve egymás utáni kódú karakterekkel (char adattag), valamint 1-től n-ig számokkal (int adattag), majd kiírja a tömb teljes tartalmát egy fájlba bináris formában egyetlen rendszerhívással. Ezután visszaolvassa az értékeket a tömbbe kettesével úgy, hogy az első kettőt az [n-2], [n-1]; a második kettőt az [n-4], [n-3]; és így tovább, míg az utolsó kettőt a [0], [1] indexű elemekbe olvassa. Pontosan a leírt sorrendben végezze az olvasást, tehát a fájlt az elejétől folytonosan olvasva tölti fel a tömböt. Végül írja ki a tömb tartalmát a képernyőre.

Példa:

n=6 esetén tömbtartalom kezdeti feltöltés után:

	[0]	[1]	[2]	[3]	[4]	[5]
char	'a'	'b'	'c'	'd'	'e'	'f'
int	1	2	3	4	5	6

tömbtartalom visszaolvasás után:

	[0]	[1]	[2]	[3]	[4]	[5]
char	'e'	'f'	'c'	'd'	'a'	'b'
int	5	6	3	4	1	2

8. Módosítsa az előző feladatban leírt programot úgy, hogy a program beolvasáskor a tömböt a [0] indexű elemtől előre haladva tölti fel, kettesével visszafelé olvasva a fájlt!

9. Készítsen programot, mely a következő három jól elkülönülő részből áll! Inicializálás: Hozza létre „szamok.dat” fájlt (ha létezett törölje és úgy hozza létre), majd deklarálja 0 kezdőértékkel x változót és írja bele a fájlba x változó értékét! Végül zárja le a fájlt!

Számolás: Nyissa meg „szamok.dat” fájlt, majd ismételje n-szer (n int típusú konstans, értéke 1000) a következőt:

olvassa be a fájlban tárolt számot,

növelje eggyel a beolvasott értéket,

írja felül a fájlban a számot a megnövelt értékkel!

Végül zárja le a fájlt!

Kiírás: Nyissa meg „szamok.dat” fájlt, olvassa ki a benne tárolt számot és írja ki a képernyőre!

10. Módosítsa az előző programot az alábbiak szerint:

- A feladat Számolás szakaszában szereplő ciklus előtt bontsa két szálra a végrehajtást!
- A szülő várja be a gyerek terminálását a Kiírás szakasz megkezdése előtt!

- A feladat Kiírás szakaszát csak a szülő folyamatban hajtsa végre!

Próbálja ki a programot úgy is, hogy a szülő nem várja be a gyerek terminálását! Figyelje meg többszöri futtatás esetén milyen jelenségeket tapasztal, és magyarázza meg ezeket!

11. Módosítsa az előző programot az alábbiak szerint:

A feladat Számolás szakaszában szereplő ciklus magjának kezdetén zárolja a fájlt, és a ciklusmag utolsó tevékenységeként oldja fel a zárolást! Tehát csak arra az időre legyen zárolva, amennyire feltétlenül szükséges (read előtt kezdődjék és write után szűnjön meg a zárolás).

12. Módosítsa az előző programot az alábbiak szerint:

A feladat Számolás szakaszában szereplő ciklus előtt kettő helyett négy szálra bontsa a végrehajtást!

Próbálja ki a programot úgy is, hogy a szülők nem várják be a gyerek terminálását! Figyelje meg többszöri közvetlen egymás után történő futtatás esetén milyen jelenségeket tapasztal, és magyarázza meg ezeket!