2. Laborgyakorlat

echo, >, >>

Az echo parancs kiírja, a mögötte álló kifejezést amennyiben nem értelmezhető kapcsoló vagy idézőjelek „”, aposztrófok „’” között áll, majd sortörést tesz. Lehetséges kapcsolója a -n, mellyel nem tesz sortörést a kiírás után. Ha nem adunk meg paraméternek se kapcsolót se szöveget akkor egy sortörést ír ki.
A > segítségével egy program kimenetét egy fájlba tehetjük (ha nem létezett a fájl létrejön, 2x kiadva egymás után a parancsot az első fájl felülíródik).
A >> segítségével egy program kimenetét egy fájlhoz hozzáfűzhetjük (ha nem létezett a fájl létrejön, többször kiadva egymás után a parancsot az utoljára kiadott parancs kimenet a fájlhoz fűződik, amennyiben az létezett).

Feladat

1. Lépjen be diak felhasználóként az első terminálon! (`<ALT>+<F1>`)
2. Váltson könyvtárat, lépjen be a /tmp könyvtárba! (`cd /tmp`)
3. Írassa ki, 'Hello world!' szöveget echo segítségével! (`echo 'hello world'`)
4. Írassa elso.txt nevű fájlba, a „Hello world!” szöveget echo segítségével! (`echo 'Hello world!' > elso.txt`)
5. Írassa masodik.txt nevű fájlba, a „Hello” szöveget echo segítségével! (`echo 'Hello' > masodik.txt`)
6. Írassa /tmp/masodik.txt nevű fájlba, a „ world!” szöveget echo segítségével! (`echo ' world!' > masodik.txt`)
7. Írassa harmadik.txt nevű fájlba, a „Hello” szöveget echo segítségével, de ne legyen sortörés a kiírás után! (`echo -n 'Hello' > harmadik.txt`)
8. Írassa /tmp/harmadik.txt nevű fájlba, a „ world!” szöveget echo segítségével! (`echo ' world!' > harmadik.txt`)

cat, tac, <, <<

A cat parancs a paraméterként megadott állományt, vagy a bemenetére kapott adatokat kiíratja. A tac mint a nevéből is látszik a cat-hez hasonlóan működik, csak fordítva íratja ki a bemenetet. A cat-nak nem csak fájl nevet adhatunk meg, hanem a bemenetére irányíthatunk fájlokat, melyeket kilistáz mintha paraméterként kapta volna. Pl.: cat < proba.txt.

Amennyiben több sornyi szöveget szeretnénk kiíratni vagy fájlbaírni, a „<<” és a cat parancsot kell használnunk, a következő módon:

```
cat << EOF
asédjféask
adsfjaéfdkj
asdfjaésdklfjaés
asdéfkjaésdkj
EOF
```

Amíg EOF nem fog állni egy külön sorban önállóan, addig írhatunk (törölni vissza sorokat

Készítette: Sinkó Gergely

nemlehet!), melyet az cat fog majd megkapni paraméterként. Ha fájlba szeretnénk tárolni a végeredményt, a következő parancsot kell kiadnunk:

```
cat > /tmp/file.txt << EOF
adsfasédjlf
asdfasdéfjha
sdafasjdfk
asdfasdkjfhalaa
sdjkfhalsj
EOF
```

Ez a file.txt be teszi amit begépeltünk.

Feladat

1. Készítsen egy több soros (legalább 40 soros, ismétlődésekkel!) állományt, mely számokat tartalmaz több sorban! (pl.:

```
cat > /tmp/szamok.txt << EOF
123
12
43
4
54
876
...
34
564
12
546
34
EOF)
```

2. Készítsen egy több soros (legalább 40 soros, ismétlődésekkel!) állományt, mely karakterláncokat tartalmaztöbb sorban! (pl.:

```
cat > /tmp/karakterek.txt << EOF
234hlk12h
heghjg123
khjjkhg
23hg4
23hg4
...
dofg98
6l5hkj
qw93e8r7
wksdfkasjh
dofg98
sdf8g79
EOF)
```


Készítette: Sinkó Gergely

more, less, |

Mivel a terminál 25 karakter magas, a 40 soros állomány túlszalad a képernyőn. A more és less parancsokkal tudjuk 25 sornál hosszabb kimeneteket lapozni. A more csak előre (space segítségével egész oldalakat, enterrel sorokat léptet előre, kilépni q-val), a less előre és hátra is tud lapozni (működik a more-nál megszokott billentyűkombinációk, valamint a pageup, pagedown, fel, le nyilak, kilépni q-val).
A „|" az úgynevezett pipe/cső. Ennek segítségével egy program kimenetét egy másik programnak adhatjuk át (pl.: less, more).

Feladat

1. Írassa ki a képernyőre a /tmp/szamok.txt állományt!
   (`cat /tmp/szamok.txt`)
2. Írassa ki fordítva a képernyőre a /tmp/karakterek.txt állományt!
   (`tac /tmp/karakterek.txt`)
3. A more segítségével lapozza a /tmp/karakterek.txt állományt!
   (`cat /tmp/karakterek.txt | more`)
4. A less segítségével lapozza a /tmp/szamok.txt állományt!
   (`cat /tmp/szamok.txt | less`)

sort, uniq, wc, tail, head

Néha szükségünk lehet, hogy a kiíratott adatokat sorba rendezzük. Erre a sort parancs szolgál, mely a bemenetként kapott adathalmazt sorba rendezi karakterek szerint. Alapértelmezetten a számok is karakterek. Ha használjuk a -n kapcsolóját akkor numerikusan rendez sorba(-n nélkül például ilyen sorrend lenne: 1,11,112,21,213,...), a -r kapcsoló hatására fordított sorrendben írja ki a kapott bemenetet. A -rn a fordított numerikus listázás. Ennek hatására a sort csökkenő sorrendben íratja ki a számokat.

Ha többször is előfordul egy kifejezés, akkor a uniq parancs eltávolítja a „felesleget" és csak egyet hagy meg.

A wc megszámolja, hogy egy szövegben, hány szó, karakter, sor található.
A sorba rendezett adatból nem biztos, hogy mindenre szükségünk van. A head alapértelmezetten az első 10 sort, míg a tail az utolsó 10 sort írja ki. Mindkettőnek megadhatunk egy -n [szám] kapcsolót. Ebben az esetben [szám] sort fog kiírni a parancs.

Feladat

1. Rendezze fordított sorrendbe a /tmp/karakterek.txt állományt!
   (`cat karakterek.txt | sort -r`)
2. Rendezze numerikus sorrendbe a /tmp/szamok.txt állományt, és távolítsa el az egyforma értékeket!
   (`cat szamok.txt | sort -n | uniq`)
3. Rendezze fordított numerikus sorrendbe a /tmp/szamok.txt állományt, és írassa ki az 5 legnagyobb értéket!
   (`cat karakterek.txt | sort -rn | head -n 5`).

Készítette: Sinkó Gergely

4. Rendezze sorrendbe a /tmp/karakterek.txt állományt, majd távolítsa el az azonos tartalmú sorokat. Az így kapott rendezett adathalmazt wc segítségével ellemezze, hány sor, szó, karakter található benne!
   (`cat /tmp/karaktere.txt | sort |uniq | wc`)

Figyeljük meg, hogy a fenti példákban, minden esetben a sortot követte a uniq parancs. Ennek oka, hogy a uniq csak az egymás után álló azonos sorokat törli. Ha ugyan az a karakterhalmaz előfordul kétszer de nem egymás után, a uniq nem törli a sorokat. Ezért célszerű minden uniq előtt sorba rendezni az adatokat.

diff

A diff bemenetként kapott két fájlt hasonlítja össze, és a különbségeket kiírja a kimenetre.

Feladat

1. Hozza létre az elso.c nevű fájlt a következő tartalommal:

```
#include <stdio.h>
main() {
printf(„Hello World!”)
}
(cat > elso.c << EOF
#include <stdio.h>
main() {
printf(„Hello World!”)
}
EOF)
```

2. Hozza létre az masodik.c nevű fájlt a következő tartalommal:

```
#include <stdio.h>
int main(void) {
printf(„Hello World!\n”);
}
(cat > masodik.c << EOF
#include <stdio.h>
int main(void) {
printf(„Hello World!\n”);
}
EOF)
```

3. Hasonlítsa össze a két fájl tartalmát diff segítségével! (`diff elso.c masodik.c`)
4. Indítsa újra a számítógépet! (`<CTRL>+<ALT>+<DEL>`)

Készítette: Sinkó Gergely