9. Mutassa be példákon keresztül a parancs csatolást, a standard ki- és bemenet átirányítását, a semlegesítő karakterek használatát. Magyarázza el a példákat részletesen.

A standard ki- és bemenet átirányítása

Ha elindítunk egy programot, akkor az három csatornát kap az operációs rendszertől:

1. standard input, vagy szabványos bemenet: sok program innen várja az adatokat; általában a billentyűzet
2. standard output, vagy szabványos kimenet: sokszor a program adatkimenete; általában a képernyő
3. standard error, vagy szabványos hibacsatorna: a program a hibaüzeneteit a szabványos hibacsatorna felé irányíthatja; általában a képernyő

Mindhárom csatorna esetén át lehet irányítani a be- vagy kimenetet:

- ha a bemenetet irányítjuk át egy állományra, akkor a program nem a billentyűzetről olvassa be az adatokat, hanem a megadott állományból
- ha a kimenetet irányítjuk át egy állományba, akkor a program kimenete nem a képernyőn jelenik meg, hanem a megadott állományban

A szabványos kimenet átirányítását a > jellel lehet elvégezni. Ebben az esetben az állomány automatikusan létrejön, vagy ha már létezik, akkor a tartalma felülírásra kerül. Ha azt szeretnénk, hogy az adott program kimenete az állomány végéhez kerüljön hozzáfűzésre, akkor a >> kettős jelet kell használni. A szabványos hibacsatorna a 2> illetve a 2>> jelek használatával irányítható át.

Példa a kimenet átirányítására az echo parancs alkalmazásával. Az echo paranccsal egy sor szöveget lehet kiíratni.

```
echo "Hello World"
```

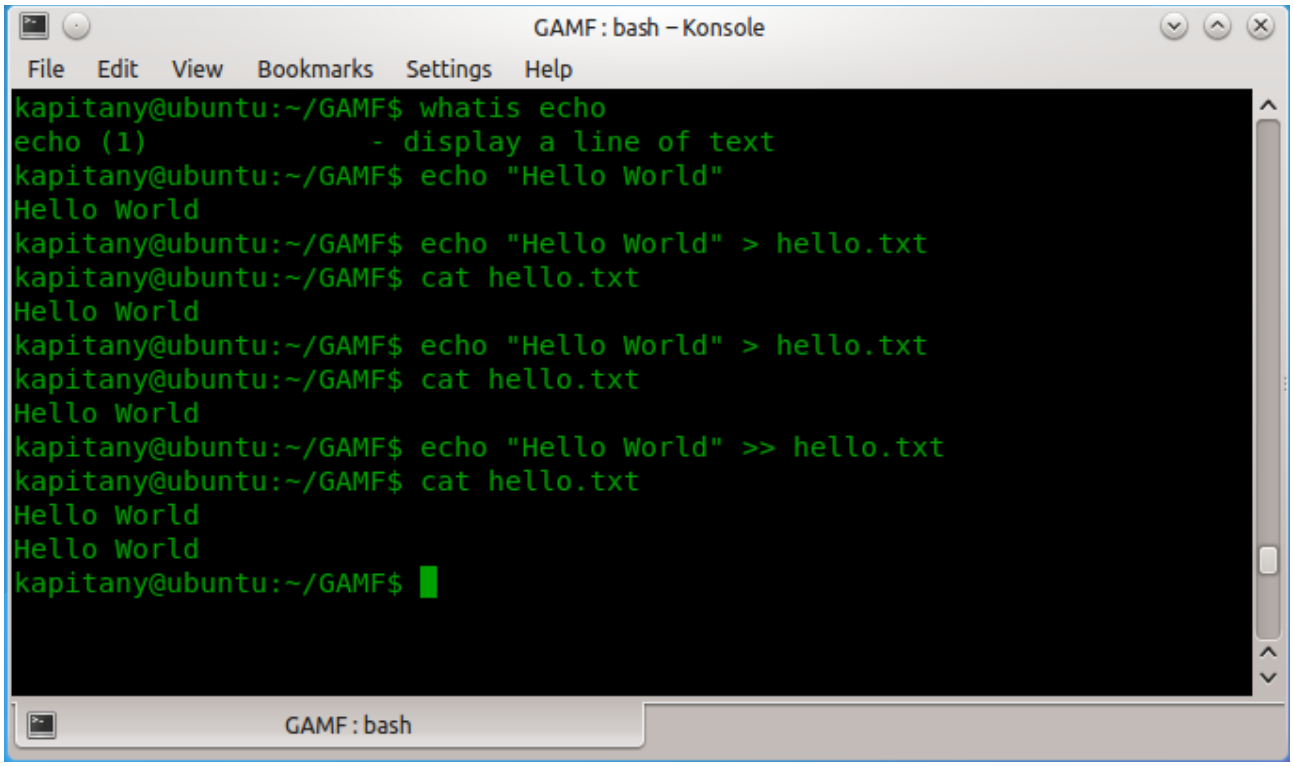
Szöveg kiírása a képernyőre.

```
echo "Hello World" > hello.txt
```

A megadott szöveg nem a képernyőre íródik ki, hanem a hello.txt nevű fájlba kerül. Ha a hello.txt már létezett, akkor tartalma felülírásra kerül.

```
echo "Hello World" >> hello.txt
```

A >> jellel történő átirányítás a megadott szöveget a hello.txt fájl végéhez fűzi hozzá.



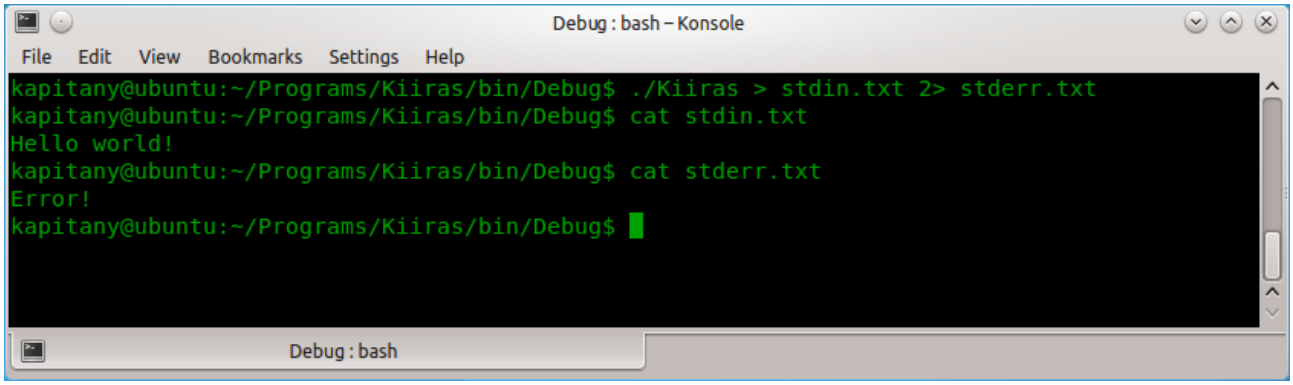
A szabványos hibacsatorna átirányításának demonstrálására készítettem egy C++ nyelvű programot, melynek a neve Kiiras:

main.cpp

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       cout << "Hello world!" << endl;
8       cerr << "Error!" << endl;
9       return 0;
10  }
11
```

Ha ezt a programot meghívjuk az alábbi paranccsal, akkor a szabványos kimenet az stdin.txt-be, a hibakimenet pedig az stderr.txt-be kerül:

```
./Kiiras > stdin.txt 2> stderr.txt
```



A szabványos bemenet átirányításának demonstrálására készítettem egy C++ nyelvű programot, melynek neve udvozles:

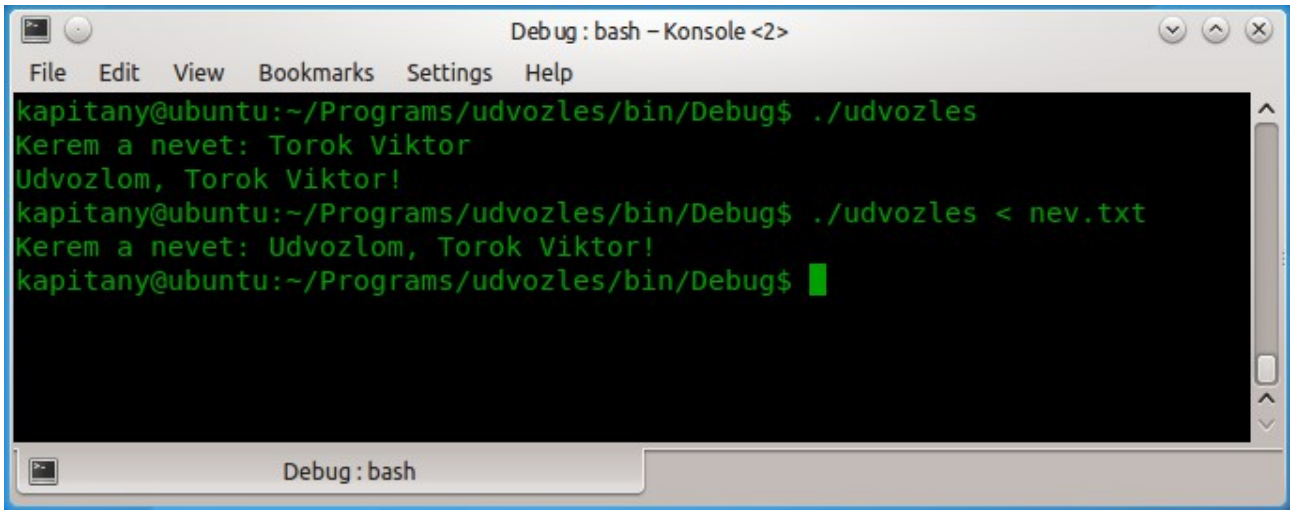
main.cpp

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main()
7  {
8      cout << "Kerem a nevet: ";
9      string nev;
10     getline(cin, nev);
11     cout << "Udvozlom, " << nev << "!" << endl;
12
13     return 0;
14 }
15
```

Ez a program alapértelmezett esetben a billentyűzetről várja a felhasználó nevét, hogy azután üdvözölhesse.

Ha viszont a következő módon hívjuk meg a programot, akkor nem a billentyűzetről olvassa be az adatokat, hanem a nev.txt nevű fájlból:

```
./udvozles < nev.txt
```



Semlegesítő karakterek

\ karakter: a mögötte álló karaktert védi; például a helyettesítő karakterek feloldására használható

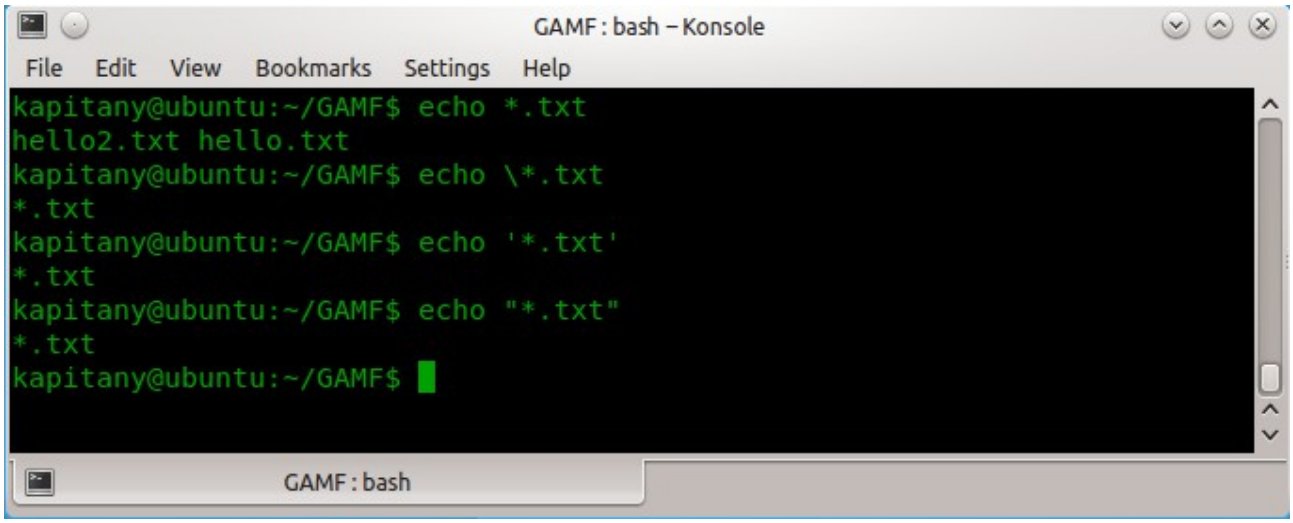
Ha a képernyőre a *.txt szöveget szeretnénk kiírni, akkor meg lehet próbálni az alábbi parancsot:

```
echo *.txt
```

Ez a parancs az eredeti szándéktól eltérően nem a *.txt szöveget írja ki a képernyőre, hanem a könyvtárban található txt kiterjesztésű fájlok nevét. A * helyettesítő karakter feloldására használható a \ jel:

```
echo \*.txt
```

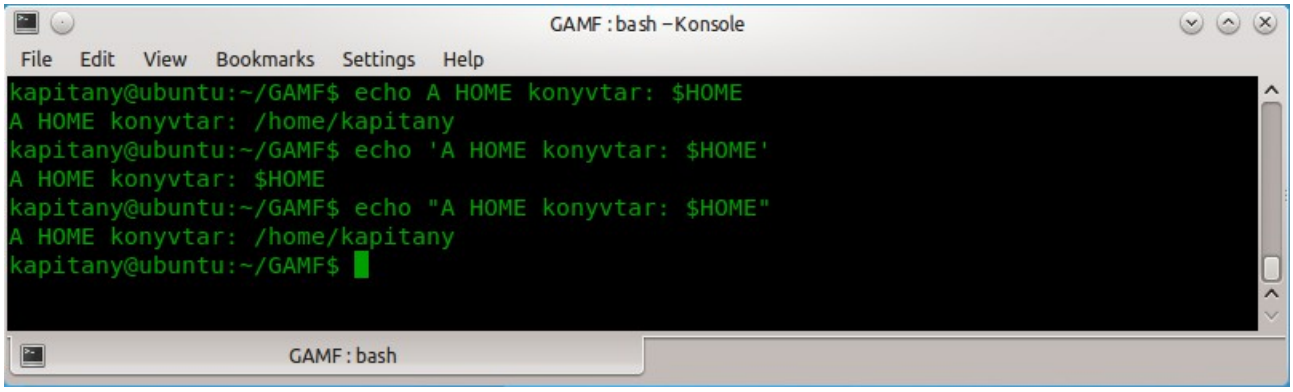
A feladat megoldható a ' vagy a " semlegesítő karakterek alkalmazásával is.



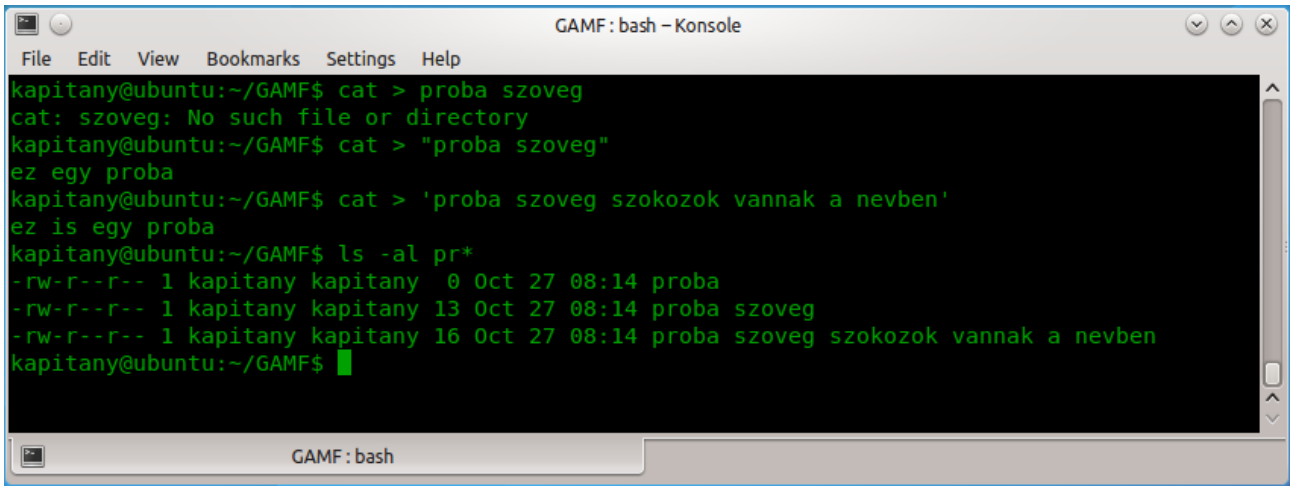
'...': shell behelyettesítéstől védi a közrefogott szöveget; a szóközök hatását megszünteti környezeti változó behelyettesítés nélkül

"…": shell behelyettesítéstől védi a közrefogott szöveget, de a környezeti változókat behelyettesíti a szövegbe

A következő képernyőképen látható, hogy a $HOME környezeti változó, illetve szöveg kiírása hogyan történik, attól függően, hogy milyen határolójeleket használunk:



A '…' és a "…" határolójelek használhatók abban az esetben is, ha egy adott fájl neve szóközöket tartalmaz:



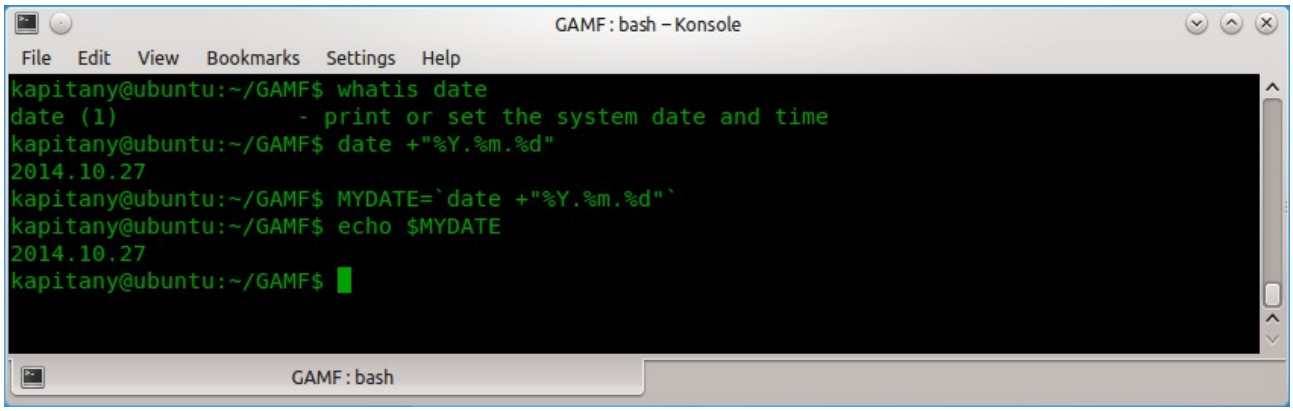
`parancs`: parancshelyettesítés; kiértékeli a közrefogott parancsot, és annak értékét helyettesíti az idézőjeles kifejezés helyére

Példa: tegyük a MYDATE nevű változóba az aktuális dátumot. Ebben az esetben a date parancsot használhatjuk a kívánt dátumformátum megadásával:

```
date +"%Y.%m.%d"
```

Ha a parancs által visszaadott értéket szeretnénk betenni a MYDATE változóba, akkor azt a következő módon tehetjük meg:

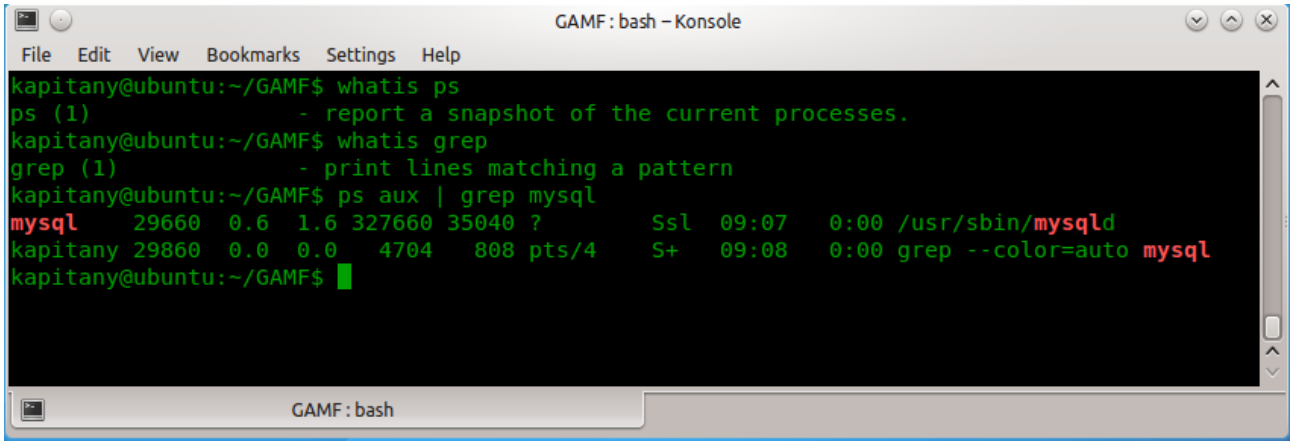
```
MYDATE=`date +"%Y.%m.%d"`
```



Parancs csatolás

A parancs csatolás használata esetén az első parancs szabványos kimenete lesz a második parancs szabványos bemenete, és így tovább, azaz az egyes programok szabványos kimenetét a következő program szabványos kimenetére csatoljuk, így láncot alkotva kettő vagy akár több program felhasználásával. Ehhez a parancsok között a pipe (|) karaktert kell használnunk.

Példa: a ps parancs kiadásával le lehet kérdezni az aktuális folyamatokat. Ha ezek közül csak azokat szeretnénk megkapni, amelyekben szerepel a mysql szöveg, akkor a `ps aux` parancs kimenetét a `grep mysql` parancs bemenetére kell irányítani, így a végeredmény csak azokat a sorokat fogja tartalmazni, amelyekben megtalálható a mysql szöveg.



Ha kipróbáljuk az előzőekben ismertetett `ps aux` parancsot további szűrés nélkül, akkor azt tapasztalhatjuk, hogy a kimenet igen hosszú lesz, és nem fér el egy képernyőn. Ennek a problémának a megoldására alkalmazhatók a lapozóprogramok, például a more és a less, amelyek lehetővé teszik a képernyőre kiírt adatfolyam megállítását vagy a less esetén az ide-oda történő lapozását.

```
ps aux | more
ps aux | less
```

A ps aux | more parancs eredménye: