

**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 kiírni.

```
echo "Hello World"
```

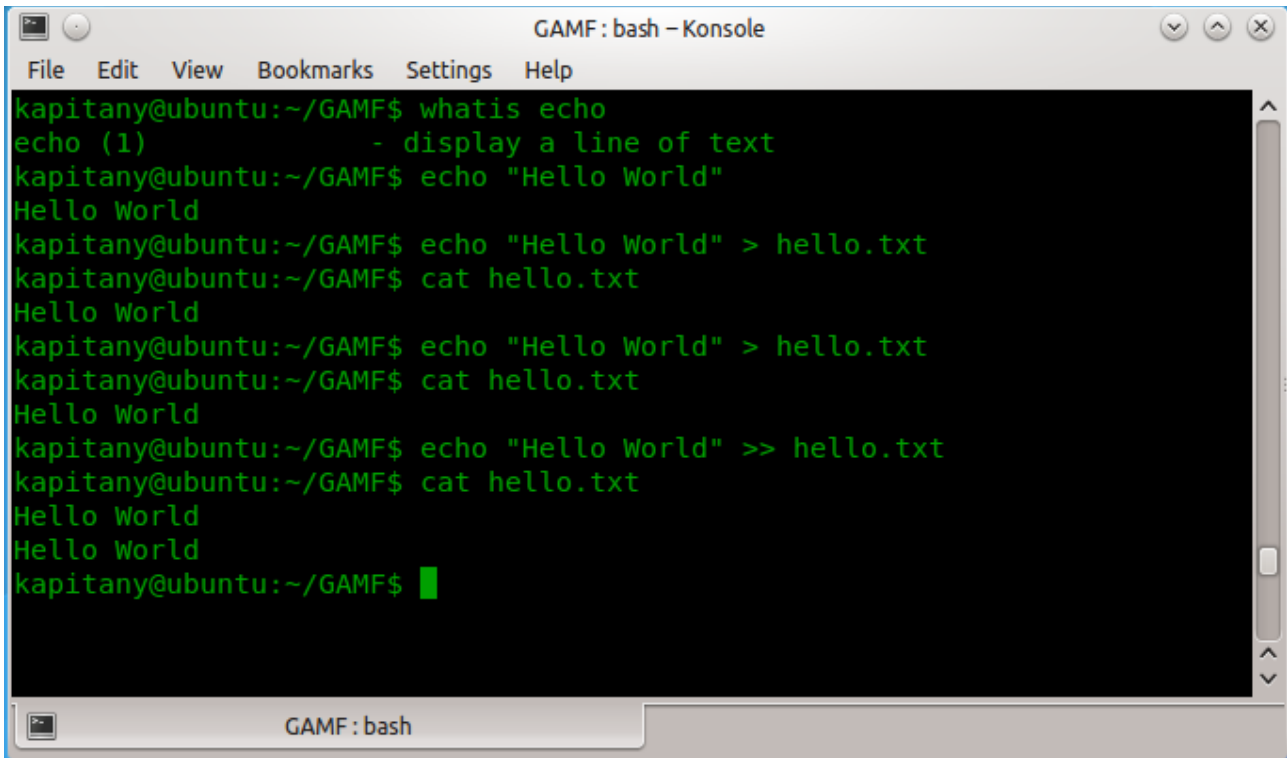
Szöveg kiírása a képernyőre.

```
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.

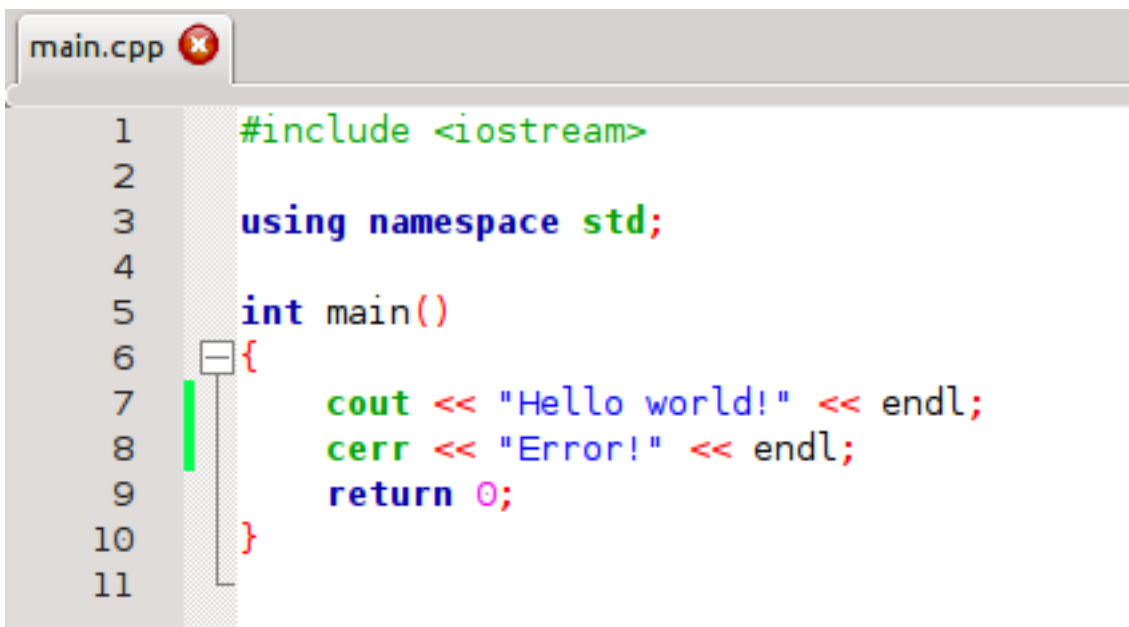
```
echo "Hello World" >> hello.txt
```

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



```
GAMF : bash - Konsole
File Edit View Bookmarks Settings Help
kapitany@ubuntu:~/GAMF$ whatis echo
echo (1)          - display a line of text
kapitany@ubuntu:~/GAMF$ echo "Hello World"
Hello World
kapitany@ubuntu:~/GAMF$ echo "Hello World" > hello.txt
kapitany@ubuntu:~/GAMF$ cat hello.txt
Hello World
kapitany@ubuntu:~/GAMF$ echo "Hello World" > hello.txt
kapitany@ubuntu:~/GAMF$ cat hello.txt
Hello World
kapitany@ubuntu:~/GAMF$ echo "Hello World" >> hello.txt
kapitany@ubuntu:~/GAMF$ cat hello.txt
Hello World
Hello World
kapitany@ubuntu:~/GAMF$ █
```

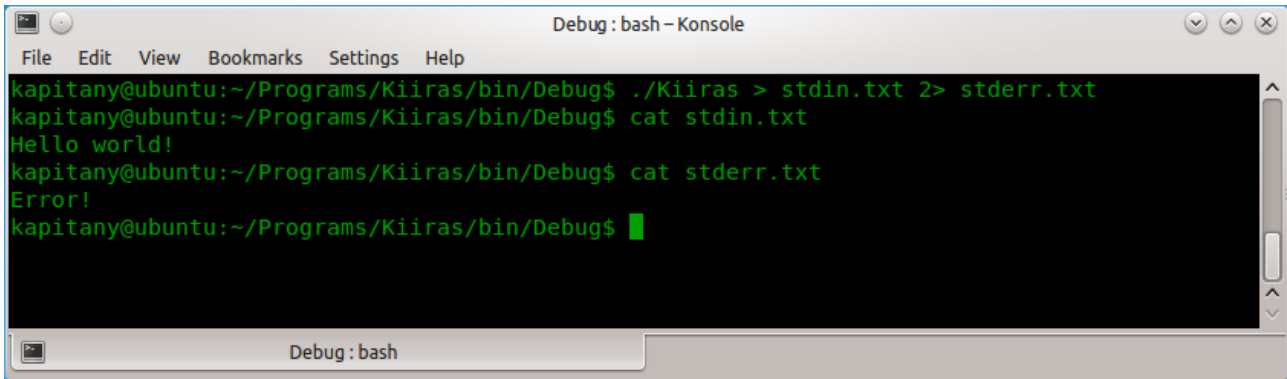
A szabványos hibacsatorna átirányításának demonstrálására készítettem egy C++ nyelvű programot, melynek a neve Kiiras:



```
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
```

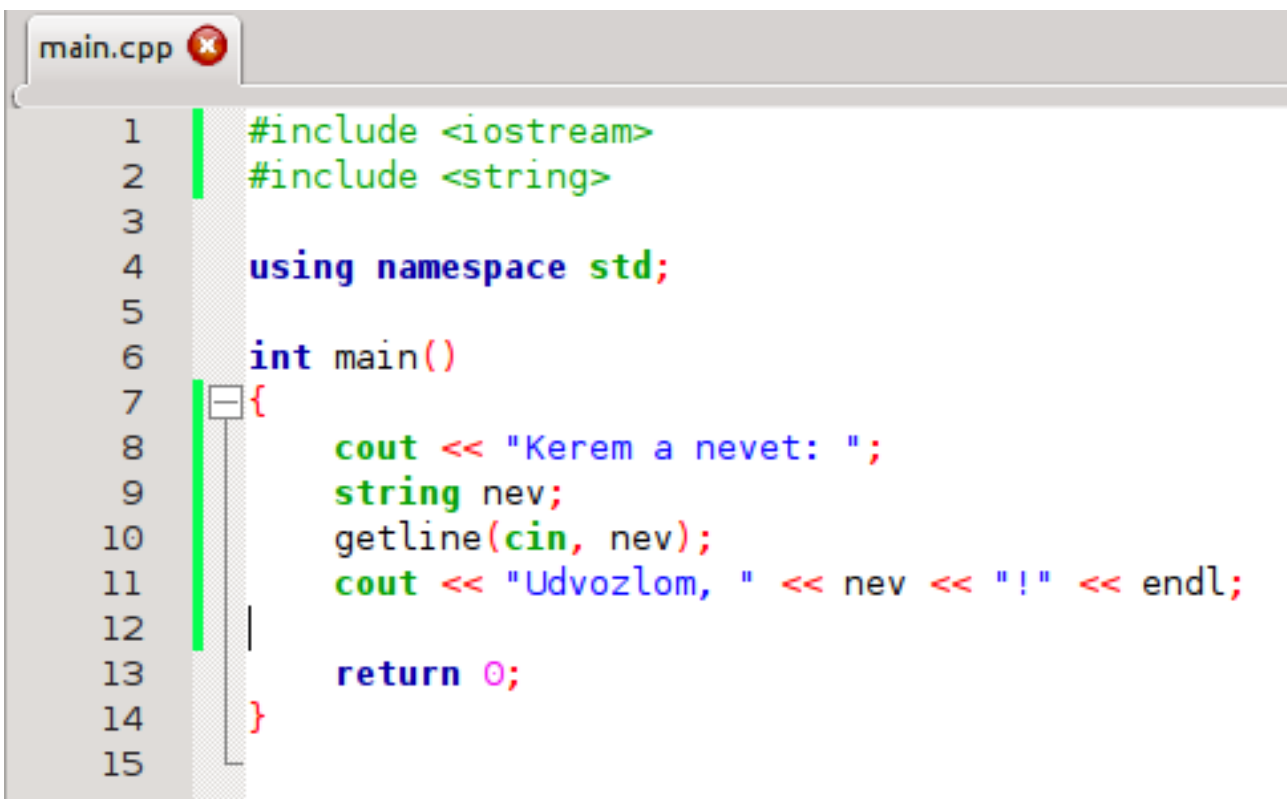
Ha ezt a programot meghívjuk az alábbi paranccsal, akkor a szabványos kimenet az stdin.txt-be, a hibakimenet pedig az stderr.txt-be kerül:

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



```
Debug : bash - Konsole
File Edit View Bookmarks Settings Help
kapitany@ubuntu:~/Programs/Kiiras/bin/Debug$ ./Kiiras > stdin.txt 2> stderr.txt
kapitany@ubuntu:~/Programs/Kiiras/bin/Debug$ cat stdin.txt
Hello world!
kapitany@ubuntu:~/Programs/Kiiras/bin/Debug$ cat stderr.txt
Error!
kapitany@ubuntu:~/Programs/Kiiras/bin/Debug$
```

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

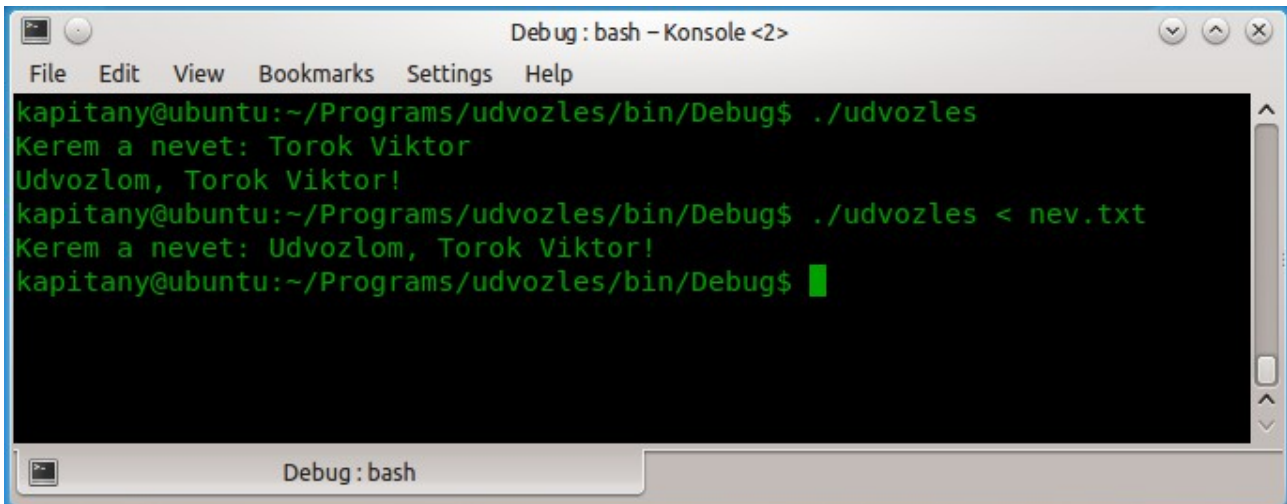


```
main.cpp
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main()
7 {
8     cout << "Kerem a nevet: ";
9     string nev;
10    getline(cin, nev);
11    cout << "Udvozlom, " << nev << "!" << endl;
12
13    return 0;
14 }
15
```

Ez a program alapértelmezett esetben a billentyűzetről várja a felhasználó nevét, hogy aztán 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
```



```
Debug : bash - Konsole <2>
File Edit View Bookmarks Settings Help
kapitany@ubuntu:~/Programs/udvozles/bin/Debug$ ./udvozles
Kerem a nevet: Torok Viktor
Udvozlom, Torok Viktor!
kapitany@ubuntu:~/Programs/udvozles/bin/Debug$ ./udvozles < nev.txt
Kerem a nevet: Udvozlom, Torok Viktor!
kapitany@ubuntu:~/Programs/udvozles/bin/Debug$ █
```

### *Semlegesítő karakterek*

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

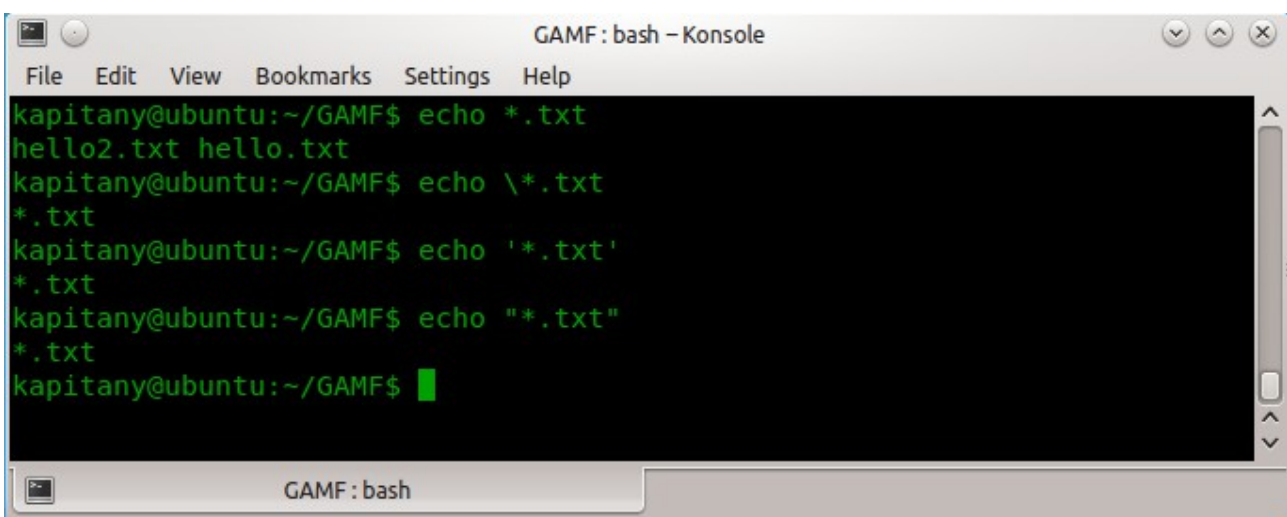
Ha a képernyőre a \*.txt szöveget szeretnénk kiírni, akkor meg lehet próbálni az alábbi parancsot:

```
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.



```
GAMF : bash - Konsole
File Edit View Bookmarks Settings Help
kapitany@ubuntu:~/GAMF$ echo *.txt
hello2.txt hello.txt
kapitany@ubuntu:~/GAMF$ echo \*.txt
*.txt
kapitany@ubuntu:~/GAMF$ echo \'*.txt\'
*.txt
kapitany@ubuntu:~/GAMF$ echo "*.txt"
*.txt
kapitany@ubuntu:~/GAMF$ █
```

'...': shell behelyettesítéstől védi a közrefogott szöveget; a szóközők 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:

```
GAMF : bash -Konzole
File Edit View Bookmarks Settings Help
kapitany@ubuntu:~/GAMF$ echo A HOME könyvtar: $HOME
A HOME könyvtar: /home/kapitany
kapitany@ubuntu:~/GAMF$ echo 'A HOME könyvtar: $HOME'
A HOME könyvtar: $HOME
kapitany@ubuntu:~/GAMF$ echo "A HOME könyvtar: $HOME"
A HOME könyvtar: /home/kapitany
kapitany@ubuntu:~/GAMF$
```

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

```
GAMF : bash -Konzole
File Edit View Bookmarks Settings Help
kapitany@ubuntu:~/GAMF$ cat > proba szoveg
cat: szoveg: No such file or directory
kapitany@ubuntu:~/GAMF$ cat > "proba szoveg"
ez egy proba
kapitany@ubuntu:~/GAMF$ cat > 'proba szoveg szokozok vannak a nevben'
ez is egy proba
kapitany@ubuntu:~/GAMF$ ls -al pr*
-rw-r--r-- 1 kapitany kapitany  0 Oct 27 08:14 proba
-rw-r--r-- 1 kapitany kapitany 13 Oct 27 08:14 proba szoveg
-rw-r--r-- 1 kapitany kapitany 16 Oct 27 08:14 proba szoveg szokozok vannak a nevben
kapitany@ubuntu:~/GAMF$
```

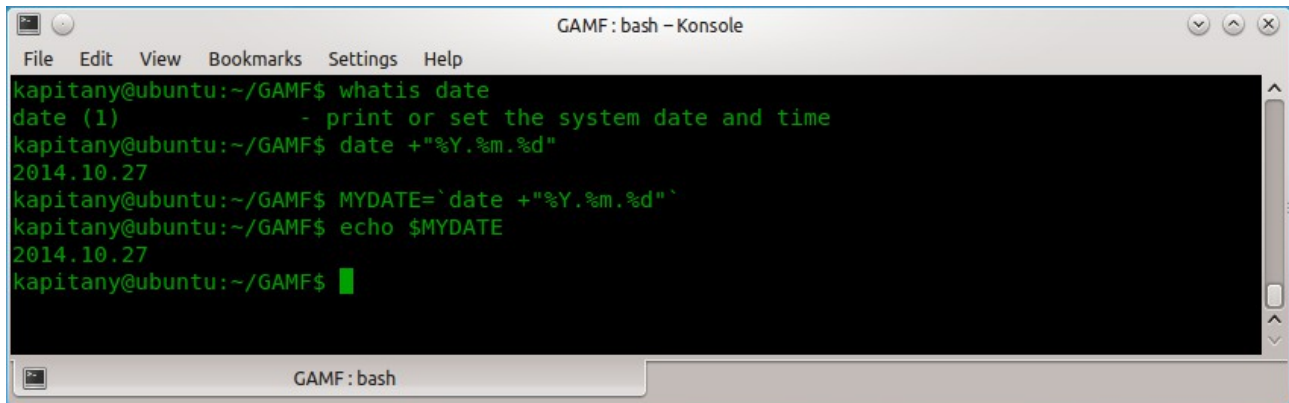
`parancs`: parancshelyettesítés; kiértékeli a közrefogott parancsot, és annak értékét helyettesíti az idézőjeles kifejezés helyére

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" `
```

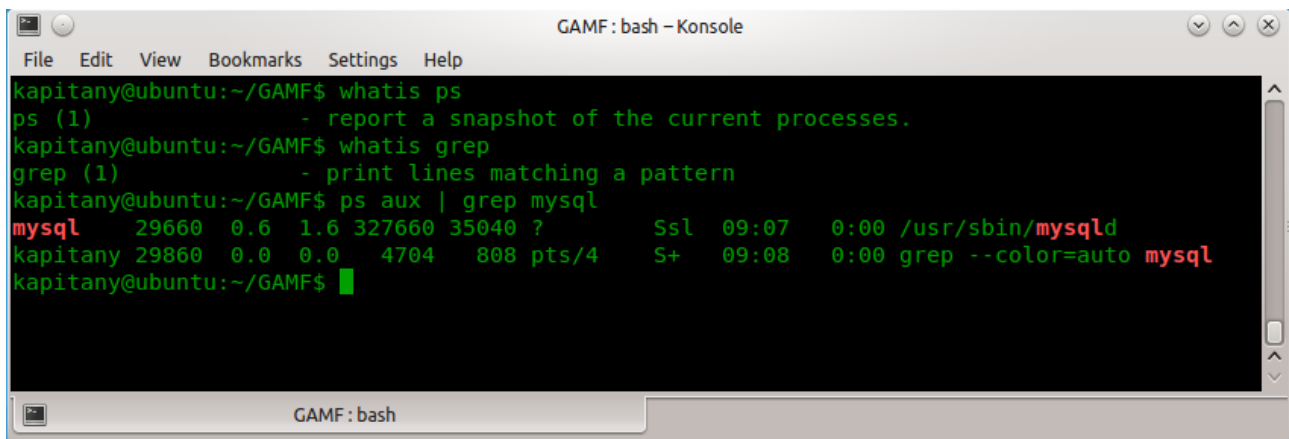


```
GAMF : bash - Konsole
File Edit View Bookmarks Settings Help
kapitany@ubuntu:~/GAMF$ whatis date
date (1) - print or set the system date and time
kapitany@ubuntu:~/GAMF$ date +"%Y.%m.%d"
2014.10.27
kapitany@ubuntu:~/GAMF$ MYDATE=`date +"%Y.%m.%d"`
kapitany@ubuntu:~/GAMF$ echo $MYDATE
2014.10.27
kapitany@ubuntu:~/GAMF$
```

### 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.

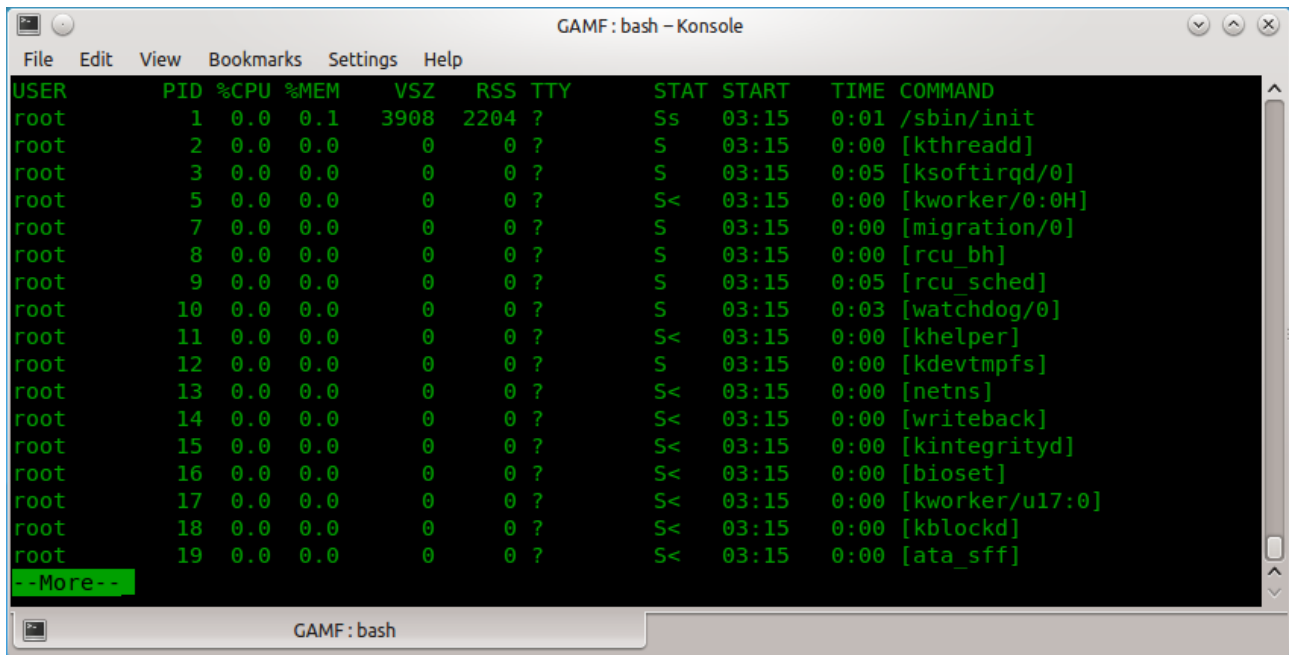


```
GAMF : bash - Konsole
File Edit View Bookmarks Settings Help
kapitany@ubuntu:~/GAMF$ whatis ps
ps (1) - report a snapshot of the current processes.
kapitany@ubuntu:~/GAMF$ whatis grep
grep (1) - print lines matching a pattern
kapitany@ubuntu:~/GAMF$ ps aux | grep mysql
mysql      29660  0.6  1.6 327660 35040 ?        Ssl  09:07   0:00 /usr/sbin/mysqld
kapitany  29860  0.0  0.0   4704   808 pts/4    S+   09:08   0:00 grep --color=auto mysql
kapitany@ubuntu:~/GAMF$
```

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:



```
GAMF: bash - Konzole
File Edit View Bookmarks Settings Help
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  3908  2204 ?        Ss   03:15   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    03:15   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    03:15   0:05 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   03:15   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    03:15   0:00 [migration/0]
root         8  0.0  0.0      0     0 ?        S    03:15   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    03:15   0:05 [rcu_sched]
root        10  0.0  0.0      0     0 ?        S    03:15   0:03 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S<   03:15   0:00 [khelper]
root        12  0.0  0.0      0     0 ?        S    03:15   0:00 [kdevtmpfs]
root        13  0.0  0.0      0     0 ?        S<   03:15   0:00 [netns]
root        14  0.0  0.0      0     0 ?        S<   03:15   0:00 [writeback]
root        15  0.0  0.0      0     0 ?        S<   03:15   0:00 [kintegrityd]
root        16  0.0  0.0      0     0 ?        S<   03:15   0:00 [bioaset]
root        17  0.0  0.0      0     0 ?        S<   03:15   0:00 [kworker/u17:0]
root        18  0.0  0.0      0     0 ?        S<   03:15   0:00 [kblockd]
root        19  0.0  0.0      0     0 ?        S<   03:15   0:00 [ata_sff]
--More--
GAMF: bash
```