7 - Átiranyítások

Alapfogalmak

stdin

A stdin tulajdonképpen a billentyzet.

stdout

Az stdout a képerny.

stderr

Az stderr szintén a képerny.

cat parancs

A cat parancs egy olyan parancs, amely billentyzetrl olvas, és stdout-ra ír.

Átirányítás fájlba vagy fájlból

Az alapértelmezett kimenet átirányítása

Most a billentyzetrl fogunk olvasni, a kimenet az stdout helyett viszont egy állomány lesz. Az átirányítást a nagyobb-mint (>) karakterrel fogjuk megvalósítani.

A gyumolcs.txt fájl tartalmát a képerny helyett a gyumolcs2.txt fájlba irányítom:

```
cat > gyumolcs.txt
szilva
barack
körte
alma<Ctrl>+<D>
```

A bevitelt a <Ctrl>+<D> billentykombinációval szakítjuk meg, amellyel tulajdonképpen „fájlvége jelet” küldünk a rendszernek.

Bármely program, ha a stdout-ra (képernyre) ír annak kimenetét átirányíthatjuk fájlba.

Az alapértelmezett hibakimenet átirányítása

Néhány program a normál üzenetek mellett hibaüzeneteket is a képernyre íródnak. Ezek a hibaüzenetek nem kerülnek fájlba az elbbi átirányítás esetén: A hibaüzenetek átirányítását is megoldhatjuk azonban:

Tegyük fel, hogy nem létezik az aktuális könyvtárban nincs „aaaa” nev könyvtár és fájl sem, amikor kiadjuk a következ parancsot:

```
ls aaaa
```

Az eredmény a képernyre íródik.

```
ls: aaaa nem érhet el: Nincs ilyen fájl vagy könyvtár
```

Ha ezt megpróbáljuk fájlba irányítani a fenti módon nem járunk sikerrel, mivel ez az az üzenet a stdout helyett a stderr-re íródik. Persze alapértelmezetten az is képerny. Az alapértelmezett kimenet száma a „2”. Meg kell mondanunk, hogy a 2-es számú kimenet szeretnénk fájlba irányítani:

```
ls aaaa 2> a.txt
```

Ezek után a képernyre nem ír semmit a parancs. Az a.txt fájlban viszont:

```
ls: aaaa nem érhet el: Nincs ilyen fájl vagy könyvtár
```

A bemenet átirányítása

A gyumolcs.txt fájl tartalmát a képernyre írom:

```
cat < gyumolcs.txt
```

A cat bemenete alapértelmezetten a billentyzet. Mi most azt mondtuk, hogy ez egy fájl legyen.

A bemenet és kimenet együttes átirányítása

Az adatokat a gyumolcs.txt fájlból vesszük. A gyumolcs.txt fájl tartalmát a képerny helyett a gyumolcs2.txt fájlba irányítom:

```
cat < gyumolcs.txt > gyumolcs2.txt
```

Ha a fájl nemlétezik akkor létrehozza. Ha létezik felülírja.

Hozzáfzés

A kimenet átirányítását a nagyobb-mint (>) karakterrel végeztük. Ez mint láttuk felülírja a létez fájlt. Néha azonban ezt szeretnénk elkerülni, és szeretnénk a már meglév fájl végéhez fzni. Ezt kett darab nagyobb-mint jel egymásután írásával érjük el:

```
echo "banán" >> gyumolcs.txt
```

A fenti parancs a gyumolcs.txt fájl végéhez fzi a „banán” szöveget.

Az stdout és az stderr együttes átirányítása

Néha szeretnénk a hibaüzeneteket és a normál üzeneteket is fájlba irányítani. Az alábbi program normál és hibaüzeneteket is generál.

```
program > kimenet 2>&1
```

vagy:

```
program &> kimenet.txt
```

A kimenet stdout és az stderr képernyre és fájlba írása, példa:

```
dir letezofaj.txt nemletezofajl.txt 2>&1 | tee kimenetifajl.txt
```

A „tee” parancs egyszerre ír fájlba és a képernyre.

A parancs után a képernyn és a kimenetifajl.txt állományban is:

```
dir: nemletezofajl.txt nem érhet el: Nincs ilyen fájl vagy könyvtár
letezofajl.txt
```

A stderr és stdout háttere

Az alapértelmezett kimenet és az alapértelmezett hibakimenet szétválasztása a programozásnál kezddik. A megértéshez vegyünk egy egyszer C programot, amely a képernyre ír. Képernyre ír például a printf() függvény:

```
printf("Egy\n");
```

Ugyanezt csinálja a következ utasítás is:

```
fprintf(stdout, "Egy\n");
```

Csak itt meg kellett adni, hogy az alapértelmezett kimenet nevét: stdout.

A következ utasítás azt mondja, hogy írjunk az alapértelmezett hibakimenetre:

```
fprintf(stderr,"Három\n");
```

A programozó tehát néha azt mondhatja, hogy írjunk a képernyre, de legyen megkülönböztetve a példában a „Három” szó kiíratása, mert az a hibakimenetre megy. A „\n” az sortörés.

Lássunk egy komplett programot:

main.c#include <stdio.h> main() { printf("Egy\n"); fprintf(stdout, "Kett\n"); fprintf(stderr, "Három\n"); }

A programban a „Egy” és „Kett” szavak az alapértelmezett kimenetre mennek. Az alapértelmezett kimenet száma 1. A „Három” szó viszont a már a hibakimenetre megy, melynek száma 2.

Ha gépünkön telepítve van a gcc fordító, akkor fordítsuk le a programot:

```
gcc -o main main.c
```

Elkészül egy main nev állomány. Ha futtatjuk:

```
./main
```

a képernyre íródik mind a három szám:

```
Egy
Kett
Három
```

Most használjuk az átirányítást. Irányítsuk fájlba az alapértelmezett kimenetet:

```
./main > naplo.txt
```

vagy:

```
./main 1> naplo.txt
```

Ekkor a „Három szót a képernyre íjra a program, az „Egy” és a „Kett” viszont a naplo.txt nev állományba kerül.

Most irányítsuk az alapértelmezett hibakimenetet a állományba:

```
./main 2> naplo.txt
```

Most a képernyre íródik az „Egy” és a „Kett” szó, a „Három” pedig állományba.

Ha pedig mind két kimenetet állományba szeretném irányítani, akkor:

```
./main &> naplo.txt
```

Ekkor mind a három szó fájlba kerül.

Láthatjuk, hogy a stdout és stderr technika lehetvé teszi, hogy elválasszuk egy program normál üzeneteit a hibaüzenetektl, ami céljainktól függen néha nagyon jól jöhet.

Átirányítás program számára

A következkben a szintén átirányítást fogunk használni, azonban most egy másik program számára adjuk át a kimenetet. Ezt a pipe (|) karakter segítségével valósítjuk meg.

```
cat gyumolcs.txt | sort
```

Többszörös átirányítás

Létrehozunk egy állományt.

```
cat > gyumolcs3.txt
körte
szilva
barack
alma
szilva<Ctrl>+<D>
```

Az állomány tartalmaz néhány ismétldést. Ezt kezelni tudjuk az uniq paranccsal. Az uniq parancs azonban csak rendezett állománnyal képes dolgozni. Elször tehát rendeznünk kell a tartalmat.

```
cat gyumolcs3.txt | sort | uniq
```

Esetleg ha szeretnénk megszámoltatni a sorok számát is, akkor:

```
cat gyumolcs3.txt | sort | uniq | wc -l
```

Az egésznek az eredményét pedig ugyancsak fájlba irányíthatjuk:

```
cat gyumolcs3.txt | sort | uniq | wc -l > gyumolcsszam.txt
```

Gyakorlat

- Mi a stdin?
- Mi a stdout?
- Mire használjuk a ”<„ karaktert?
- Mire használjuk a ”>„ karaktert?
- Mire használjuk a „|” karaktert?
- Írassuk a ki a belépett felhasználók neveit, a kimenetet a users.txt fájlba irányítva.