

1. Készítsen programot, melyben szülő vár, míg gyerek küld neki egy szignált, majd kiírja, hogy "Szülő vége!"; gyerek pedig vár 2 mp-t, majd küld egy SIGQUIT szignált szülőnek, végül kiírja, hogy "Gyerek vége!". Ebben a programban ne használjon még handlert, és figyelje meg, hogy a gyerektől kapott szignál így kilövi szülőt, tehát a „Szülő vége!” nem íródik ki.
2. Az 1. feladatban elkészített programot más szignálok használatával is próbálja ki (pl. SIGCHLD, SIGSTOP, SIGTERM, SIGKILL)! Figyelje meg a szignált fogadó folyamat reakciója és a küldött szignál alapértelmezett tevékenysége (ld. man) közötti összefüggéseket!
3. Az 1. feladatban elkészített programot módosítsa úgy, hogy SIG_IGN használatával figyelmen kívül hagyja a szülőnek küldött szignált! Figyelje meg a program viselkedését! Miben tér el az 1. feladatban elkészített program viselkedésétől?
4. Az 1. feladatban elkészített programot módosítsa úgy, hogy egy handlert (kezelő függvény) kapcsol szülőben a küldött szignálhoz! A handler legyen egy „üres” függvény. Figyelje meg a program viselkedését! Miben tér el az 1. feladatban elkészített program viselkedésétől?
5. Készítsen programot, melyben gyerek 1 másodpercnyi várakozást követően küldjön SIGTERM szignált szülőnek, erre válaszként szülő azonnal küldjön SIGUSR1 szignált gyereknek! Mindkét folyamat írja ki, ha megkapta a szignált! Végül szülő várja meg gyerek terminálását.
6. Készítsen programot két gyerekfolyamattal. Mindkét gyerek várjon egy szignál érkezésére, majd írja ki a saját nevével (Gyerek1 vagy Gyerek2) kezdve a sort, hogy szignál érkezett hozzá, végül szintén a saját nevével kezdve a sort, írja ki, hogy véget ért és termináljon. Szülő várjon 3 másodpercet, majd küldjön egy SIGUSR1 szignált minden vele egy csoportban lévő folyamatnak egyetlen kill használatával (vagyis 0-ás PID-re küldje), majd ezt követően várja meg minden gyerek terminálását. Végül írja ki, hogy „Szülő folyamat véget ért.” Az összes folyamatban a SIGUSR1 szignált ugyanaz a handler kezelje, mely kiírja, hogy hányas számú szignál érkezett, és mi a szignál jelentése.

Minta:

```
10 számú szignál érkezett
A szignál jelentése: User defined signal 1
10 számú szignál érkezett
10 számú szignál érkezett
A szignál jelentése: User defined signal 1
A szignál jelentése: User defined signal 1
Gyerek2 szignál érkezett.
Gyerek1 szignál érkezett.
Gyerek2 folyamat véget ért.
Gyerek1 folyamat véget ért.
Szülő folyamat véget ért.
```

Figyelje meg, hogy a handler hányszor kerül végrehajtásra, és magyarázza meg, miért annyiszor!

7. Módosítsa a 6. feladatban leírt programot úgy, hogy a szülő nem a vele egy csoportban lévő folyamatoknak küldi a szignált, hanem minden olyan folyamatnak elküldi, amely felé szignálküldési engedéllyel rendelkezik (vagyis -1-es PID-re küldi)! Figyelje meg, mi történik, ha futtatja, és magyarázza meg, amit tapasztalt!

8. Készítsen alkalmazást két gyerekfolyamattal. Az egyik gyerekfolyamat küldjön egy SIGUSR1 szignált a szülőnek. Szülő írja ki, hogy megkapta és azonnal küldjön egy SIGUSR2 szignált a másik gyereknek, aki fogadja azt és szintén kiírja, hogy megkapta!

9. Indítson egy programból pontosan két folyamatot. Az egyik írja bele egy fájlba `argv[1]`-et, a másik pedig olvassa ki és írja ki a képernyőre (másiknak célszerű megvárnia, amíg az egyik végez).

10. Hozzon létre azonos szülőfolyamatból két gyerekfolyamatot (*gy1* és *gy2*).

- A *gy1* gyerekfolyamat végtelen ciklusban, 0,2 másodpercenként írjon konzolablakba egy sort mely egyetlen # karaktert tartalmaz.
- A *gy2* gyerekfolyamat várjon 3 másodpercet, majd egy SIGSTOP szignállal függessze fel *gy1* működését. Újabb 3 másodperc várakozás után egy SIGCONT szignállal indítsa el ismét *gy1*-et. Végül további 3 másodperccel később SIGTERM-mel termináltassa *gy1*-et és maga is termináljon.
- A *szülő* a gyerekek létrehozását követően várjon *gy1* felfüggesztésére, és ha az bekövetkezik, tájékoztassa róla a felhasználót! Ezután *gy1* folytatására várjon, és ennek bekövetkezéséről is tájékoztasson. Végül az összes gyerekfolyamat terminálását követően a szülő is termináljon!

11. Optimalizálja az `fcntl.c` mintaprogramot! Minden kódrész csak egyszer szerepeljen. Ne ismétlődjön ugyanaz a kód a szülő és a gyerek folyamatokban.

12. Írja át `fcntl.c` mintaprogramot úgy, hogy mindegyik folyamat a zárolás során `F_SETLK` módot használjon `F_SETLKW` helyett! Miben tér el az így kapott `data.txt` tartalma az `F_SETLCKW` használatával előállított `data.txt`-től? Magyarázza el az eltérés okát!

13. Írja át `fcntl.c` mintaprogramot úgy, hogy csak az egyik folyamatban zárolja a fájlt írásra! Miben tér el az így kapott `data.txt` tartalma az `fcntl.c` használatával előállított `data.txt`-től? Magyarázza el az eltérés okát!

14. A gyerekfolyamat hozzon létre egy fájlt, zárolja írásra, majd 1 másodpercnyi várakozást követően küldjön egy szignált a szülőnek. Ezután írja ki karakterenként 20 μ sec várakozással egy fájlba a „Gyerek vagyok Gyerek vagyok...” szöveget 25-ször. Végül oldja fel a zárolást. A szülő várjon egy szignál érkezésére, majd zárolja a gyerek által létrehozott fájlt olvasásra és listázza ki karakterenként a fájl tartalmát a konzolablakba.

15. Készítsen programot, mely SIGKILL szignállal kilövi a parancsértelmezőt, amiből elindítjuk (bezárja Putty-t).