

A feladatok megoldása során a jövőben NE használja a `pause`, a `sleep` és a `usleep` függvényeket! A folyamatok felesleges várakozás nélkül, a lehető leghatékonyabban fussanak le. Az időzítéseket időzítővel oldja meg!

1. Készítsen programot, mely

- A közös kódban blokkol minden szignált, majd indít egy gyerekfolyamatot.
- A szülő várakozzon 10 db SIGUSR1 szignál érkezésére, majd egy SIGTERM szignál küldésével termináltassa gyereket.
- Gyerek egy végtelen ciklusban ismételje a következőket:
 - alarm használatával várakozzon 1 másodpercet, majd
 - küldjön egy SIGUSR1 szignált szülőnek véletlenszerűen döntve arról, hogy `kill`-el vagy `sigqueue`-val küldi. Ha `sigqueue`-val küldi, küldjön vele egy véletlenszámot is.
 - Írja ki, hogy mit küldött (azonosítható legyen, hogy a gyerek írja).
- A handler írja ki - szintén azonosítható módon -, amit kapott.

```
Gyerek: kill ->
                Handler: kill
Gyerek: kill ->
                Handler: kill
Gyerek: sigqueue 1451032874 ->
                Handler: sigqueue 1451032874
Gyerek: kill ->
                Handler: kill
Gyerek: sigqueue 824043833 ->
                Handler: sigqueue 824043833
Gyerek: kill ->
                Handler: kill
Gyerek: kill ->
                Handler: kill
Gyerek: sigqueue 1834056131 ->
                Handler: sigqueue 1834056131
Gyerek: kill ->
                Handler: kill
Gyerek: kill ->
                Handler: kill
```

2. Készítsen programot, mely 3 paramétert kap. Az első paraméter az első időzítés, a második paraméter az ismételt időzítés értéke legyen másodpercben megadva, a harmadik paraméter pedig az események (ismétlések) száma legyen. Állítsa be az ismételt időzítőt a paramétereknek megfelelően (ne feledje a szignálok maszkolását), és az időzítő minden lejáratakor írja ki a minta alapján, hogy még mennyi idő van hátra. Végül állítsa le az időzítőt, és írja ki, hogy „Vége”.

```
klettner@szamrend:~$ a.out 10 1 12
Még 12 * 1 másodperc van hátra!
Még 11 * 1 másodperc van hátra!
Még 10 * 1 másodperc van hátra!
Még 9 * 1 másodperc van hátra!
Még 8 * 1 másodperc van hátra!
Még 7 * 1 másodperc van hátra!
Még 6 * 1 másodperc van hátra!
Még 5 * 1 másodperc van hátra!
Még 4 * 1 másodperc van hátra!
Még 3 * 1 másodperc van hátra!
Még 2 * 1 másodperc van hátra!
Még 1 * 1 másodperc van hátra!
Vége!
```

Próbálja ki a programot úgy is, hogy az időzítések valamelyikéhez 0 értéket ad meg!

3. Hozzon létre egy programban három időzítőt 2, 3 és 5 másodperces ismétlődő időzítésekre! Mindhárom időzítő lejáratkor küldjön egy SIGUSR1 szignált, és a szignállal küldje el az időzítés értékét is (pl. az 5 másodperces az 5-ös értéket). Hozzon létre továbbá egy egyszeri 30 másodperces időzítőt is, mely a lejáratkor egy SIGTERM szignállal termináltatja a programot. Indítsa el mind a négy időzítőt, majd a program egy végtelen ciklusban várakozzon.

A 2, 3 és 5 másodperces időzítők lejáratkor írja ki a program a képernyőre, hogy melyik járt le éppen.

```
klettner@szamrend:~$ a.out
2 másodpercenkénti időzítő lejárt!
3 másodpercenkénti időzítő lejárt!
2 másodpercenkénti időzítő lejárt!
5 másodpercenkénti időzítő lejárt!
2 másodpercenkénti időzítő lejárt!
3 másodpercenkénti időzítő lejárt!
2 másodpercenkénti időzítő lejárt!
3 másodpercenkénti időzítő lejárt!
5 másodpercenkénti időzítő lejárt!
2 másodpercenkénti időzítő lejárt!
3 másodpercenkénti időzítő lejárt!
2 másodpercenkénti időzítő lejárt!
Befejeződött
klettner@szamrend:~$
```

4. Módosítsa a 3. feladatban leírt programot úgy, hogy SIGUSR1 helyett SIGRTMIN szignált használ! Figyelje meg, miben tér el a két program futása egymástól!

5. Készítsen programot, mely 25 másodpercen keresztül számlál vissza R (véletlenszerű 1-8 közé eső érték) másodpercenként. Az időzítő úgy működjön, hogy az utolsó R másodperces várakozás leteltekor teljen le a 25 másodperc is. Ezt úgy oldja meg, hogy szükség esetén az első várakozás legyen rövidebb R másodpercnél. Pl. R=7 esetén az első időzítés 4 mp legyen, így 4+7+7+7 ütemezéssel pont 25 másodpercnél ér véget az utolsó 7 másodperces időzítés. R értékének kisorsolását követően tájékoztassa a felhasználót az ütemezésről a mintának megfelelően!

Az időzítő minden lejáratkor küldjön egy SIGUSR1 szignált, melyet egy handler kezeljen le. A handler a mintának megfelelően írja ki minden alkalommal, hogy épp hány másodpercenkénti visszaszámlálás (R értéke) hatására fut le.

A fentieket ismételje meg 3 alkalommal!

A szignálra való várakozáshoz használhat pause-t.

```
klettner@os:~/feladatok/6feladatsor> a.out
25 másodperc visszaszámlálás 4 másodpercenként (1 mp, majd 6 db 4 mp)
24
4 másodpercenkénti visszaszámlálás!
20
4 másodpercenkénti visszaszámlálás!
16
4 másodpercenkénti visszaszámlálás!
12
4 másodpercenkénti visszaszámlálás!
8
4 másodpercenkénti visszaszámlálás!
4
4 másodpercenkénti visszaszámlálás!
25 másodperc visszaszámlálás 3 másodpercenként (1 mp, majd 8 db 3 mp)
24
3 másodpercenkénti visszaszámlálás!
21
3 másodpercenkénti visszaszámlálás!
18
3 másodpercenkénti visszaszámlálás!
15
3 másodpercenkénti visszaszámlálás!
12
3 másodpercenkénti visszaszámlálás!
9
3 másodpercenkénti visszaszámlálás!
6
3 másodpercenkénti visszaszámlálás!
3
3 másodpercenkénti visszaszámlálás!
25 másodperc visszaszámlálás 5 másodpercenként (0 mp, majd 5 db 5 mp)
25
5 másodpercenkénti visszaszámlálás!
20
5 másodpercenkénti visszaszámlálás!
15
5 másodpercenkénti visszaszámlálás!
10
5 másodpercenkénti visszaszámlálás!
5
5 másodpercenkénti visszaszámlálás!
klettner@os:~/feladatok/6feladatsor>
```

6. Készítsen programot, mely indít egy gyerekfolyamatot.

- Gyerek 5 alkalommal 5 másodpercenként (itt használhat sleep-et) küld a szülőnek egy SIGUSR1 szignált, és a szignállal elküldi, hányadik szignált küldi éppen. Végül újabb 5 másodperc várakozás után küldjön szülőnek egy SIGUSR2 szignált.
- Szülő folyamatosan írja ki a pontos időt, ugyanazon sorba, azt a hatást keltve, mintha járna az óra, mindaddig, míg egy SIGUSR2 szignál nem érkezik hozzá. A SIGUSR2 érkezése után írjon egy soremelést a képernyőre, végül termináljon.
Ha miközben írogatja az időt, érkezik egy SIGUSR1 szignál, akkor írja ki egy új sorba a szignállal küldött számot (azt, hogy hányadik alkalommal érkezett SIGUSR1 szignál). Minden kiírást a szülő végezzen, a handler ne írjon semmit a képernyőre!

```
klettner@szamrend:~$ a.out
22:57:46
1
22:57:51
2
22:57:56
3
22:58:01
4
22:58:06
5
22:58:11
klettner@szamrend:~$
```

7. Készítsen programot, mely indít két gyerekfolyamatot.

- Mindkét gyerek véletlenszerűen kiválasztott mennyiségű (1-10) db egyjegyű nem negatív véletlenszámot sorsoljon. Minden sorsolást követően írja ki a kisorsolt számot és a saját PID-jét a képernyőre, majd küldje el a kisorsolt számot a szülőnek egy szignálhoz kapcsolva. Végül küldjön egy negatív értéket egy szignálhoz kapcsolva.
- Szülő adja össze a gyerekektől kapott számokat, és amikor mindkét gyerektől megkapta a jelzést, hogy végzett (negatív érték), akkor írja ki a kapott számok összegét.

```
4 (PID: 12703)
3 (PID: 12704)
8 (PID: 12703)
0 (PID: 12703)
1 (PID: 12704)
1 (PID: 12703)
6 (PID: 12704)
9 (PID: 12704)
7 (PID: 12704)
A számok összege: 39
```

8. Készítsen programot, mely szimulálja azt a helyzetet, amikor a szülő szétosztja a feladatokat a gyerekek között, miközben ő maga is dolgozik valami feladat megoldásán. A gyerekek közelítéssel számolják ki a nekik átadott N db szám gyökét, miközben a szülő egy számláló futtatásával van elfoglalva. A gyerekek a szülő felé SIGRTMAX szignállal jelezzék, hogy készen állnak a következő feladat fogadására, SIGRTMIN-nel pedig, hogy végeztek a feladattal, és a számítás eredményét is ehhez a szignálhoz csatolva adják át. Szülő SIGUSR1 szignálhoz csatolva adja át a jelentkező gyerekek az értéket, melynek a gyökét ki kell számítani, és SIGUSR2-vel jelezze, ha nincs több feladat, terminálhat a gyerek.

A konkrét megoldáshoz indítson két gyerek folyamatot.

- Mindkét gyerek küldjön SIGRTMAX szignált a szülőnek, jelezve, hogy készen áll a következő számítás elvégzésére, majd várjon, egy szignálra. Ha SIGUSR1 szignál érkezik, akkor a következő közelítéssel határozza meg a szignállal kapott egész szám (x) gyökét: 1-től indulva egyesével növelje a gyök értékét mindaddig míg olyan y értéket nem kap, melyre igaz, hogy y négyzetének távolsága x -től kisebb, mint $y+1$ négyzetének távolsága. A számításhoz ne használjon kész matematikai függvényeket, az abszolútértéket is elágazással határozza meg (a cél épp az, hogy a feladat megoldása egyrészt időigényes legyen, másrészt változó mértékű időre legyen szükség az egyes konkrét esetekben a megoldásához)! Az így kapott y érték egy olyan egész szám, mely x gyökéhez a legközelebb áll. A fenti közelítő számítás eredményét egy SIGRTMIN szignálhoz kapcsolva küldje el szülőnek! A fentieket addig ismételje, míg egy SIGUSR2 szignált nem kap. A SIGUSR2 hatására termináljon!
- Szülő kérje be N értékét, majd hozzon létre egy *szamok* és egy *gyokok* azonosítójú N elemű tömböt. Ezután indítson egy számlálót, mely a képernyő alsó sorában fut folyamatosan. A számláló írja ki az aktuális értékét (folyamatosan egyesével növekedjen) és a még hátra lévő gyökszámítások számát. Miközben a számláló fut, figyelje, hogy érkezik-e szignál (de ne várakozzon rá, tehát ne akassza meg a számlálót).
Ha SIGTRMAX szignál érkezik, sorsoljon egy véletlen számot és helyezze el a *szamok* tömbben, majd a kisorsolt számot küldje el egy SIGUSR1 szignálhoz kapcsolva annak a folyamatnak, akitől a SIGTRMAX-ot kapta.
Ha SIGRTMIN szignál érkezik, akkor az ahhoz kapcsoltan érkező számot helyezze el a *gyokok* tömbbe!
Amikor mind az N db szám gyökét megkapta, akkor állítsa le a számlálót, jelezze SIGUSR2 szignálok elküldésével a gyerekeknek, hogy nincs több feladat, majd listázza ki a *szamok* és a *gyokok* tömbök tartalmát, végül a gyerekek terminálását követően termináljon a szülő is.

Futtassa a programot kis (~ 10) és nagy (~ 100000) N értékekkel is!

Könnyen belátható, hogy a fenti program nem tökéletes, hiszen a *gyokok* tömb i . elemébe nem feltétlen a *szamok* tömb i . elemének gyöke kerül, de az biztos, hogy a *szamok* tömb minden elemének gyöke megtalálható a *gyokok* tömbben. Ennek megoldása a feladat célja szempontjából lényegtelen bonyolítás lett volna, de szorgalomból megoldhatja.

N=10				
számláló:		571 feldolgozandó: 1		
Számok:				
829127045	1312750414	1960959957	456376097	1942525984
1383973218	1276431054	785853357	1630822501	2122950708
Gyökök:				
41907	40383	36232	44283	21363
44074	37202	35727	28033	46075

9. Módosítsa a 8. feladatban leírt programot úgy, hogy a szülő mérje a számláló indítása és leállítása között eltelt időt, és a tömbök kiíratását követően írja ki, hány másodpercig tartott a gyökök kiszámítása!

10. Módosítsa a 9. feladatban leírt programot úgy, hogy kettő helyett öt gyerekfolyamat végezze a számításokat!

Hasonlítsa össze a 9. és 10. program feldolgozási sebességét! Érdeemes kipróbálni a következőket:

- azonos ($N \geq 100000$) értékekkel próbálni egymás után többször futtatni mindkettőt,
- azonos ($N \geq 100000$) értékekkel próbálni egyidőben (két putty) többször futtatni mindkettőt,
- eltérő (2:5 arány) értékekkel próbálni egyidőben (két putty) többször futtatni mindkettőt.