Feladat

Készítsünk programot, mely kiírja a saját PID-jét, majd végtelen ciklusban várakozik!

```
#include <stdio.h>
int main()
{
        printf("PID=%i\n", getpid());
        while(1);
}
```

Feladat

Indítsuk el a programot, majd egy másik terminálablakból küldjünk neki:

- SIGCHLD
- SIGUSR1
- SIGUSR2
- SIGTERM
- SIGKILL

szignálokat!

Handler

Alap handler függvény:

```
void handler(int signum)
{
        ...
}
```

Handler

signal({signal}, {handler})

A szignálhoz rendeli a kezelő eljárást.

{handler}:

- a kezelő eljárás belépési pontja, vagy
- SIG_IGN: figyelmen kívül hagyja, vagy
- SIG_DFL: alapértelmezett tevékenységet kapcsolja be.

psignal({signal}, {saját hibaüzenet})

Kiírja {saját hibaüzenet} szövegét, majd kettőspont és szóköz után a szignál leírását.

Feladat

Módosítsuk az előző programot úgy, hogy a SIGCHILD, SIGUSR1 és SIGTERM szignálok érkezésekor tájékoztassa a felhasználót, milyen szignál érkezett, de ne terminálján a program. Ha SIGUSR2 vagy SIGKILL érkezik, azt hagyja figyelmen kívül!

Indítsuk el a programot, és küldjük el neki a fenti szignálokat egy másik terminálablakból!

Feladat

```
#include <stdio.h>
#include <signal.h>

void handler(int signum)
{
        psignal(signum,"Szignál érkezett");
}

int main()
{
        signal(SIGCHLD, handler);
        signal(SIGUSR1, handler);
        signal(SIGTERM, handler);
        signal(SIGUSR2, SIG_IGN);
        signal(SIGKILL, SIG_IGN);
        printf("PID=%i\n", getpid());
        while(1);
}
```

Szignálok blokkolása (maszkolás)

<signal.h>

- **sigset_t**: szignálkészlet típus.
- **sigemptyset**(&sigset)
  Inicializálja és üressé teszi a szignálkészletet.
- **sigfillset**(&sigset)
  Inicializálja és hozzáadja az összes szignált a szignálkészlethez.
- **sigaddset**(&sigset, {szignál})
  Hozzáadja {szignál}-t a szignálkészlethez.
- **sigdelset**(&sigset), {szignál})
  Eltávolítja {szignál}-t a szignálkészletből.

Szignálok blokkolása (maszkolás)

- **sigprocmask**({művelet}, &sigset, &oldset) szignálok blokkolásának beállítása.

  {művelet}:

  - SIG_BLOCK: Hozzáadja a sigset-ben található szignálokat a blokkolt szignálokhoz
  - SIG_UNBLOCK: Eltávolítja a sigset-ben található szignálokat a blokkolt szignálok közül
  - SIG_SETMASK: A sigset-ben található szignálok lesznek a blokkolt szignálok

Feladat

Módosítsuk az előző programot úgy, hogy a PID kiírását követően maszkoljuk a SIGUSR1, SIGUSR2 és SIGTERM szignálokat, majd pause használatával várjunk egy szignál érkezésére, ezután pedig oldjuk fel a szignálok maszkolását, és csak ezt követően kerüljön sor a végtelen ciklusban való várakozásra!

Indítsuk el a programot, és egy másik terminálablakból küldjünk neki (ebben a sorrendben) SIGUSR1, SIGUSR2, SIGTERM, SIGCHLD és SIGKILL szignálokat!

Feladat

```
printf("PID=%i\n", getpid());
sigset_t sigset;
sigemptyset(&sigset);
sigaddset(&sigset, SIGUSR1);
sigaddset(&sigset, SIGUSR2);
sigaddset(&sigset, SIGTERM);
sigprocmask(SIG_SETMASK, &sigset, NULL);
pause();
sigprocmask(SIG_UNBLOCK, &sigset, NULL);
while(1);
```

Várakozás szignálra

- **sigsuspend**(&sigset)
  A folyamat szignál maszkját átmenetileg lecseréli a paraméterként átadott szignálkészletben megadottra, majd felfüggeszti a folyamat végrehajtását amíg egy olyan nem maszkolt szignál érkezik, mely meghív egy handlert vagy terminálja a folyamatot.
  - ha a szignál terminálja a folymatot, a sigsuspend nem tér vissza (értelemszerűen, hiszen a folyamat terminált).
  - ha a szignál kezelése megtörtént (handler végzett), akkor a folyamat szignál maszkja visszakapja sigsuspend előtti értékét, és a folyamat végrehajtása folytatódik.

Valós idejű (real-time) szignálok

Ha egy blokkolt szignál többször is küldésre kerül a blokkolás alatt, akkor a blokkolás feloldásakor:

- ha nem real-time szignál, akkor csak egyszer érkezik meg,
- ha real-time szignál, akkor sorban áll, és mind megérkezik.

A real-time szignálok SIGRTMIN és SIGRTMAX közé eső szignálok.

rt library használata

A real-time funkciókat tartalmazó programok fordításakor meg kell adni, hogy linkeljen az rt library-vel. Ezt a fordítónak a -lrt kapcsoló megadásával jelezhetjük.

Pl. gcc valami.c -lrt

Fájl zárolás

<fcntl.h>

- struct **flock**
  - l_type (short) lezárás típusa:
    - F_RDLCK – olvasásra,
    - F_WRLCK – írásra,
    - F_UNLCK – feloldás.
  - l_whence (short) a lock-olás kezdőpozíciójának viszonyítási pontja:
    - SEEK_SET – fájl elejétől,
    - SEEK_CUR – aktuális pozíciótól,
    - SEEK_END – fájl végétől.
  - l_start (off_t) a lock-olás kezdőpozíciója.
  - l_len (off_t) a lock-olt bájtok száma, 0: teljes fájl.
  - l_pid (pid_t) a lock-oló folyamat PID-je.
  - …

Fájl zárolás

fcntl({fileazonosító}, {mód}, &flock)

{mód}:

- F_SETLKW – beállítja a zárolást (flock szerint), vár, ha már zárolva van.
- F_SETLK – beállítja a zárolást, nem vár.