

Operációs rendszerek

7. gyakorlat

Információ küldése szignállal

Időzítés



Problémák

1. Ki küldte a szignált? (pid)
2. Mi generálta a szignált? (kill? Időzítő?,...)
3. Milyen esemény készítette a küldőt a szignál elküldésére?
(pl. SIGFPE: Nullával osztott? Túlcsondult?
Érvénytelen művelet? ...)
4. Milyen információt akar közölni a szignál segítségével a küldő?
(SIGUSR1 és SIGUSR2 kevés lehet.)

union típus

A **union** egy olyan típus, mely lehetővé teszi különböző típusú adatok tárolását a memória ugyanazon helyén.

Deklaráció:

```
union azonosító {  
    típus1 adattag1;  
    típus2 adattag2;  
    ...  
}
```

Az union típusú változó memóriafoglalása akkora, amekkora a legnagyobb méretű adattag memóriaigénye.

gy2/sigaction_value.c

Összetett (három paraméteres) handler függvény:

```
void handler(int signo, siginfo_t *info, void *context)
{
    ...
}
```

struct **siginfo_t** *info:

- info->si_code: Mi generálta a szignált? *(ld. Problémák 2.)*
 - SI_USER: kill küldte,
 - SI_QUEUE: sigqueue küldte (ld. később),
 - SI_TIMER: timer küldte (ld. később),
 - SI_MESGQ: üzenetsor küldte (ld. később),
 - ...
- info->si_pid: A küldő PID-je. *(ld. Problémák 1.)*
- info->si_value: A szignállal küldött információ. *(ld. Problémák 3., 4.)*
 union sival_t típusú (int vagy pointer):
 - info->si_value.sival_int: egy int típusú érték,
 - info->si_value.sival_ptr: egy mutató.

Összetett handler kapcsolása szignálhoz

- struct **sigaction**

Eseményleíró struktúra. Adattagjai:

- sa_sigaction: kezelő függvény belépési pontjának mutatója, vagy SIG_DFL, SIG_IGN valamelyike.
- sa_mask: sigset_t típusú, a kezelő függvény végrehajtása közben *további* blokkolandó szignálok készlete.
- sa_flags: jelzések, melyek a szignál viselkedését határozzák meg:
 - SA_SIGINFO: melyik handler-t használja?
 - ha nincs beállítva: void handler(int signo)
 - ha be van állítva: void handler(int signo, siginfo_t *info, void *context)
- ...

Handler kapcsolása szignálhoz

- **sigaction:** Egy szignál hatására bekövetkező eseményt az eseményleíró struktúrában meghatározott értékek szerint állítja be.



Információ küldése szignállal

- **sigqueue**: szignált küld egy folyamatnak, és a szignállal küld egy union sigval típusú értéket is.



Problémák

- A szignál érkezéséről nem tud az a folyamat, amihez a szignált küldik. Ahhoz, hogy a címzett folyamat tudomást szerezzen a neki küldött szignálról, fel kell függeszteni a folyamatot egy szignál érkezéséig.
- A szignállal küldött információ a handlerhez érkezik, nem ahhoz a folyamathoz, melynek a szignált címezték. Így ez az információ nem jut el a címzett folyamathoz.
- A szignállal küldött információt akkor kell feldolgozni, amikor érkezik, vagyis a küldő folyamat határozza meg a feldolgozás idejét, nem a fogadó.

Szignál kezelése folyamatból



- **sigwaitinfo**(&*szignálkészlet*, &*info*)
- **sigtimedwait**(&*szignálkészlet*, &*info*,
&*várakozás*)

Várakozik míg

- (sigtimedwait) lejár a *várakozás* által meghatározott idő, vagy
- érkezik a folyamathoz egy a *szignálkészlet*-ben szereplő szignál. Ekkor *info* ugyanúgy kap értéket, ahogy az összetett handler *info* paramétere.

Értéke: Az érkezett szignál száma, vagy
-1, ha nem érkezett szignál az időtúllépésig.

gy2/setitimer.c

Egyszeri időzítés:

Időzítő (intervall timer): szignált küld amikor az időzítő lejár, de a folyamat végrehajtását nem függeszti fel.

Egyszeri időzítés:

- **alarm**(*másodperc*)

A megadott idő leteltekor küld egy SIGALRM szignált.

- struct **timeval**
 - tv_sec: másodperc,
 - tv_usec: milliomod (mikro) másodperc.

A két érték összege jelent egy időintervallumot.
- struct **itimerval**
 - it_interval (timeval):
ismételt időzítés értéke
(ha 0, akkor egyszeri időzítés),
 - it_value (timeval):
lejáratig hátralévő idő $\equiv$ első időzítés értéke
(ha 0, akkor leállítja az időzítőt).

- **setitimer**(*időzítő*, &*új timer*, &*régi timer*)
Beállítja és elindítja az ismétlődő időzítőt.

időzítő:

- ITIMER_REAL: a ténylegesen eltelt időt méri, és lejáratkor SIGALRM szignált küld,
- ITIMER_VIRTUAL: az időzítőt indító folyamat végrehajtásának idejét méri, és lejáratkor SIGVTALARM szignált küld.
- ...
- **getitimer**(*időzítő*, &*timer*)
Lekérdezi az *időzítő* által meghatározott időzítés értékét.

Tetszőleges szignál és érték küldése időzítővel:

- **timer_create**(*clock id*, &*signal event*, &*timer id*)
Létrehoz egy új időzítőt és beállítja annak viselkedését.

clock id:

- CLOCK_REALTIME: beállítható valós idejű óra
- ...

signal event (sigevent típusú struktúra):

- sigev_notify: értesítési mód (int):
 - SIGEV_NONE: nem történik semmi az esemény bekövetkezésakor
 - SIGEV_SIGNAL: szignált küld az esemény bekövetkezésekor
 - ...
- sigev_signo: szignál, amit küld (int),
- sigev_value: a szignállal küldött union sigval tip. információ
- ...

timer id (timer_t): a létrehozott timer azonosítója

Ismétlődő időzítés

- struct **timespec**
 - tv_sec: másodperc,
 - tv_nsec: milliárdod (nano) másodperc.
- struct **itimerspec**
 - it_interval (timespec): ismételt időzítés értéke,
 - it_value (timespec): lejáratig hátralévő idő
=első időzítés értéke.

Tetszőleges szignál és érték küldése időzítővel

- **timer_settime**(*timer id*, *flags*, &*új timer*, &*régi timer*)
Beállítja és elindítja a *timer id* azonosítójú időzítőt az itimerspec típusú *új timer* értékének megfelelően.
- **timer_gettime**(*timer id*, &*aktuális érték*)
Lekérdezi *timer id* azonosítójú időzítőt.
- **timer_delete**(*timer id*)
Törli a *timer id* azonosítójú időzítőt.