Sygnały

Systemy Operacyjne 2

Piotr Zierhoffer

14 listopada 2011

Sygnały

Sygnał

- przerwanie programowe
- komunikat wysyłany między procesami
- informuje o zdarzeniu
- obsługiwany **asynchronicznie**
- obsługiwany przez wydzielone funkcje (ang. *signal handlers*)
- domyślne funkcje obsługi

Informacje

Lista sygnałów: `kill -l`
Opis sygnałów: `man 7 signal`

Obsługa sygnałów

Możliwe reakcje na sygnał:

- Term — zabij proces
- Ign — ignoruj sygnał
- Core — zabij proces i zrzuć obraz pamięci
- Stop — zatrzymaj proces
- Cont — wznów zatrzymany proces
- przechwycenie sygnału

Wysłanie sygnału

- Funkcja:
  `int kill(pid_t pid, int sig)`
- Parametry:
  - `pid` — identyfikator docelowego procesu
    - `pid > 0` — proces o identyfikatorze `pid`
    - `pid = 0` — wszystkie procesy z grupy, do której należy wysyłający
    - `pid = -1` — wszystkie procesy, do których się da, bez `init` i samego siebie
    - `pid < -1` — wszystkie procesy z grupy `-pid`
  - `sig` — numer sygnału
    - `sig = 0` — sygnał nie jest wysyłany, ale sprawdzane są ew. błędy
- Wartość zwracana:
  - -1 w wypadku błędu lub 0

Obsługa sygnału

- Funkcja:
  `sighandler_t signal(int signum, sighandler_t handler)`
- Parametry:
  - `signum` — numer sygnału do obsłużenia
  - `handler` — wskaźnik do funkcji obsługującej sygnał
    - `SIG_IGN` — zignoruj sygnał
    - `SIG_DFL` — użyj domyślnej funkcji obsługi
    - własna funkcja
- Wartość zwracana:
  - wskaźnik na poprzednio ustawioną funkcję (lub `SIG_IGN`, `SIG_DFL`)
  - `SIG_ERR` w wypadku błędu

Obsługa sygnału

- Uwagi:
  - nie można przechwytywać ani ignorować SIGKILL i SIGSTOP
  - ignorowanie SIGCHLD powoduje, że potomkowie nie zamienią się w zombie
- Wskaźnik na funkcję
  - `typedef void (*sighandler_t)(int);`

Stwórz typ `sighandler_t` jako wskaźnik do funkcji przyjmującej jeden parametr `int` i zwracający `void`.

  - podajemy nazwę funkcji

Przykład obsługi sygnału

```
void moj_handler(int sig_num){
     printf("Sygnał %d!\n", sig_num);
}

signal(SIGINT, moj_handler);  //ustaw własną funkcję obsługi

void (*f)(int sig);
f = signal(SIGINT, SIG_IGN); //ignoruj, zapisując w f wskaźnik
      na poprzednią funkcję

if(f != SIG_IGN && f != SIG_DFL)
{
     signal(SIGINT, f);  //przywróć własną funkcję, jeżeli
          była wcześniej ustawiona
}

signal(SIGINT, SIG_DFL);  //ustaw domyślną reakcję na sygnał
```

Oczekiwanie na sygnał

- Funkcja:
  `int pause(void)`
- Działanie
  - oczekuje na odbiór jakiegokolwiek sygnału
  - funkcja ignoruje sygnały z SIG_IGN
  - często kojarzona z sygnałem SIGALRM
- Wartość zwracana:
  - -1 po otrzymaniu sygnału, ustawia errno na EINTR

Budzik

- Funkcja:
  `unsigned int alarm(unsigned int seconds)`
- Parametry:
  - `seconds` — ilość sekund, po których wysłany ma być `SIGALRM`
- Wartość zwracana:
  - 0, jeżeli żaden alarm nie jest aktualnie planowany
  - ilość sekund, która pozostała do wysłania alarmu ustawionego poprzednim wywołaniem funkcji
- Uwagi:
  - poprzednie niewysłane alarmy są odrzucane
  - `sleep(3) may be implemented using SIGALRM; mixing calls to alarm() and sleep(3) is a bad idea.`

Ostatnie uwagi

- Obsługa sygnałów jest asynchroniczna!
- Otrzymanie nowego sygnału w trakcie wykonywania funkcji obsługi sygnału może spowodować trudne w analizie zachowanie programu.
- Obsługa sygnału nie powinna mieć nadmiernych efektów ubocznych dla aplikacji (modyfikacja globalnych zmiennych, modyfikacje logiki itd.).
- Sygnały mogą być maskowane, blokowane, wznawiane — szczegóły w `man 7 signal`.
- Przykład na stronie.