シグナル

田浦健次朗

シグナルとは

- イベント通知のための API
- イベント通知 → シグナルの発生 (配達)
  - CPU に対する割り込みの「ソフトウェア (プロセス) 版」
- どんなイベントに対してシグナルが発生する?
  - エラー, 例外的事象
    - Segmentation Fault も実はその一つ
  - 時刻の経過など
  - 明示的な送信 (kill システムコール, kill コマンド)
  - etc.

シグナルを受け取る方法

- 発生したシグナルを受け取るいくつかの方法
  1. 登録しておいた関数 (シグナルハンドラ) が呼ばれる (sigaction)
  2. ファイルディスクリプタにデータが到着する (signalfd)
  3. シグナルの到着を待つ関数 (sigwaitinfo) に返り値が返される
- デフォルトの動作 (何も登録されていない場合) はシグナルにより異なり,
  - プロセスが強制終了される
  - 無視される
- ブロックする関数 (read など) はシグナルを受け取るとリターンするものが多い (ブロックしていてもシグナルに気付けるように)

シグナルの例

青字が特によく使う・見かけるもの

- SIGINT：interrupt (典型的には端末で Ctrl-C を叩くと発生)
- SIGILL：不正命令の実行
- SIGSEGV：不正なメモリのアクセス (Segmentation Fault)
- SIGTERM：プロセスの強制終了のためのシグナル (処理可能)
  - kill コマンドがデフォルトで送信するシグナル
- SIGKILL：プロセスの強制終了のためのシグナル (処理不可能)
  - kill -9/-KILL が送信するシグナル
- SIGALRM：時間経過によって発生 (alarm, setitimer システムコール)
- SIGXCPU：CPU 使用量超過 (処理不可能)
- SIGUSER1, 2：明示的送信のための (自由に利用可能な) シグナル

シグナルの処理方法 (sigaction)

- `int sigaction(`*sig*`,` *act*`,` *oldact*`);`
  - `struct sigaction *` *act*;
  - シグナル *sig* を受信した時の動作を *act* で指定
- `struct sigaction` の中身

```
1  struct sigaction {
2    ...
3    /* 呼ばれる関数を指定するフィールド */
4    void (*sa_sigaction)(int, siginfo_t *, void *);
5    sigset_t sa_mask;
6    int sa_flags;
7      ...
8  };
```

sigaction 利用のテンプレート

```
1  /* シグナルハンドラ
2     sig を受け取ったときに行う動作 */
3  void sigint_action(int sig, siginfo_t * info, void * arg) {
4    ...
5  }
6
7
8  int main() {
9    ...
10
11   /* sig を受け取ったら sigint_action を呼ぶように設定 */
12   struct sigaction act;
13   act.sa_sigaction = sigint_action;
14   sigemptyset(&act.sa_mask);
15   act.sa_flags = SA_SIGINFO;
16   if (sigaction(sig, &act, 0) == -1) err(1, "sigaction");
17
18   ...
19 }
```

sigaction が設定されたシグナルが配達されたときの動作

- あるスレッド (*) で, (ちょうど割り込みが起きたのと似たように) 指定されたハンドラが実行される
- ハンドラが終了すると, 続きが実行される
- (*) あるスレッドはシグナルの種類により,
  - そのプロセス中の (OS が適当に選んだ)1 つのスレッド
  - シグナルの原因となる命令を実行したスレッド (SIGSEGV, SIGILL など)

Segmentation Fault もシグナルの一つ

- Segmentation Fault = 不正なメモリアクセス時に発生
  - 割り当てられていないアドレスをアクセスした
  - 割り当てられてはいるが, 保護属性 (読み・書き・実行) で許可されていないアクセスが行われた
- 関連システムコール
  - `mprotect(`*addr*`, `*len*`, `*prot*`);`
  - `mmap(`*addr*`, `*len*`, `*prot*`, `*flags*`, `*fd*`, `*offs*`);`
  - *prot* — `PROT_READ`, `PROT_WRITE`, `PROT_EXEC` (それらの bit 和)
- 例えば `PROT_WRITE` が設定されていない領域に書き込みを行うと, そのスレッドに `SIGSEGV` シグナルが配達される
- シグナルハンドラを設定していなければ, プロセスが終了する (デフォルトのアクション)