Лекция 28
Сигналы, часть 2

Системно-зависимые особенности signal(2)

- На время обработки (выполнения обработчика сигнала) повторный вызов текущего обработчика может блокироваться или не блокироваться
- Обработчик может сбрасываться после запуска или не сбрасываться
- Некоторые системные вызовы (read, write, accept...) могут перезапускаться или завершаться с errno == EINTR
- Вызов sigaction(2) позволяет управлять этим

Обработчики сигналов

- В обработчиках сигналов можно использовать только асинхронно-безопасные (async signal safe) стандартные функции
- Большинство системных вызовов (включая fork и exec) — асинхронно-безопасные
- Функции работы с динамической памятью (new, delete, malloc, free), функции работы с потоками (fopen, fprintf, <<) - **не асинхронно-безопасные**

Безопасная обработка сигналов

- Безопаснее всего в обработчике сигнала устанавливать глобальный флаг поступления сигнала, который обрабатывать в основной программе
- Для этого требуются доп. средства управления сигналами

```
volatile sig_atomic_t sigint_flag;
void hnd(int s)
{
  sigint_flag = 1;
}
```

Volatile, sig_atomic_t

- Ключевое слово volatile обозначает, что значение переменной может измениться «неожиданно» для компилятора программы
- Компилятор не должен пытаться оптимизировать обращения к переменной (например, загружая ее на регистр)
- Тип sig_atomic_t — это целый тип (обычно int), для которого гарантируется атомарная запись и чтение

Множества сигналов

// очистка множества

```
void sigemptyset(sigset_t *pset);
```

// заполнение множества

```
void sigfillset(sigset_t *pset);
```

// добавление сигнала в множества

```
void sigaddset(sigset_t *pset, int signo);
```

// удаление сигнала из множества

```
void sigdelset(sigset_t *pset, int signo);
```

Блокирование сигналов

- Если сигнал заблокирован, его доставка процессу откладывается до момента разблокирования

```
int sigprocmask(int how, const sigset_t *set,
                sigset_t *oldset);
```

- SIG_BLOCK — добавить сигналы к множеству блокируемых
- SIG_UNBLOCK — убрать сигналы из множества блокируемых
- SIG_SETMASK — установить множество

Ожидание поступления сигнала

```
int sigsuspend(const sigset_t *mask);
```

- На время ожидания сигнала выставляется множество блокируемых сигналов mask
- После доставки сигнала восстанавливается текущее множество блокируемых сигналов

Отображение сигналов на файловые дескрипторы

- Системный вызов signalfd(2) позволяет создать файловый дескриптор, работая с которым можно получать уведомления о поступлении сигналов
  - signalfd — создает файловый дескриптор
  - select/poll/epoll — ожидание события (прихода сигнала)
  - read — ожидание прихода сигнала и получения информации о нем
  - close — закрытие

Стратегия корректной работы с сигналами

- Функции-обработчики сигналов устанавливают флаг поступления сигнала
- Программа выполняется с заблокированными сигналами
- Сигналы разблокируются только на время ожидания прихода сигнала (с помощью sigsuspend или pselect) или используется signalfd а сигналы не нужно разблокировать