

# Сигналы часть 2

АКОС. Лекция №15

# Сигнал

- Отправляется процессу другим процессом или ядром
- Возможная реакция процесса:
  - игнорировать
  - завершить свою работу
  - завершить свою работу и сделать core dump
  - приостановить работу
  - возобновить работу
  - выполнить дополнительные действия

# Сигналы

Номер	Имя	По умолчанию	Описание
1	SIGHUP	Term	обрыв соединения
2	SIGINT	Term	Ctrl+C
3	SIGQUIT	Core	Ctrl+\
4	SIGILL	Core	плохая инструкция
6	SIGABRT	Core	abort()

**man 7 signal**

# Обработчик сигнала

- Может быть вызван в произвольный момент времени
  - использует текущий стек (но можно в `sigaction` указать использование произвольного адреса)
  - не должен использовать высокоуровневые функции

# Маска сигналов, ожидающих доставки

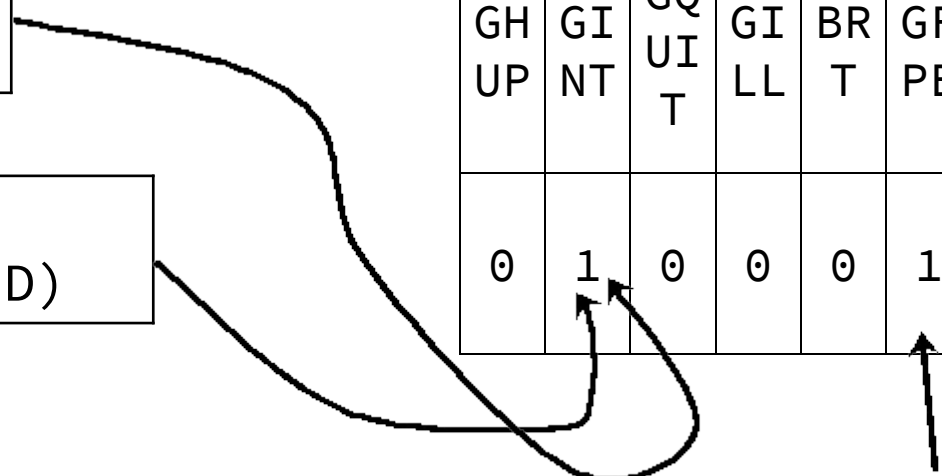
- Один из атрибутов процесса
- Не наследуется при клонировании системным вызовом `fork`

**Process 1**  
`kill(SIGINT, PID)`

**Process 2**  
`kill(SIGINT, PID)`

**Process 3**  
`kill(SIGFPE, PID)`

SI GH UP	SI GI NT	SI GQ UIT	SI GI LL	SI GA BR T	SI GF PE	SI GK ILL
0	1	0	0	0	1	0



# Доставка сигнала

- Произвольный процесс или ядро устанавливают нужный флаг в маске другого процесса
- Выполнение продолжается до тех пор, пока не планировщик не выберет другой процесс

# Доставка сигнала

- Когда планировщик заданий добирается до того процесса, который получил сигнал, то первым делом проверяется маска сигналов, ожидающих доставки

# Маска сигналов, ожидающих доставки

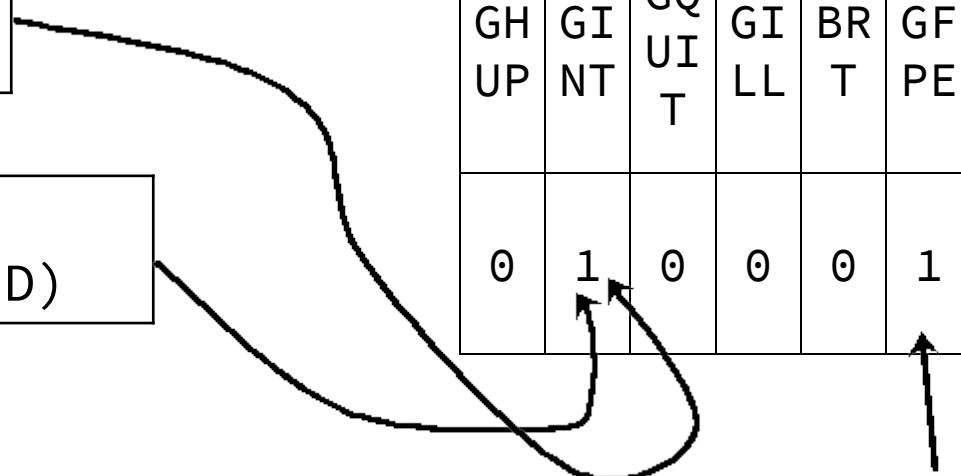
**Учитывается только факт наличия сигнала,  
но не их количество!**

**Process 1**  
`kill(SIGINT, PID)`

**Process 2**  
`kill(SIGINT, PID)`

**Process 3**  
`kill(SIGFPE, PID)`

SIGHUP	SIGINT	SIGQUIT	SIGILL	SIGABRT	SIGFPE	SIGKILL
0	1	0	0	0	1	0





# Маски сигналов

(у каждого thread'a - своя)

Какие сигналы будут обработаны →

Маска [заблокированных] сигналов →

0	0	0	0	0	1	0
1	0	1	1	1	1	1
SIGHUP	SIGINT	SIGQUIT	SIGILL	SIGABRT	SIGFPE	SIGKILL
0	1	0	0	0	1	0

**Process 1**

`kill(SIGINT, PID)`

**Process 2**

`kill(SIGINT, PID)`

**Process 3**

`kill(SIGFPE, PID)`

**Маски сигналов,**  
в отличии от маски ожидающих доставки,  
**наследуются при fork**

# Изменение маски сигналов

- Установка маски для процесса  
`sigprocmask(int how,  
              const sigset_t *mask,  
              sigset_t *old_mask)`
- Установки маски для отдельной нити  
(если используется многопоточность)  
`pthread_sigmask(int how,  
                 const sigset_t *mask,  
                 sigset_t *old_mask)`

# Множества сигналов

- Стандартом не регламентируется содержимое `sigset_t`
- Для инициализации и модификации используются функции (man sigsetops)
  - `sigemptyset`
  - `sigfillset`
  - `sigaddset`
  - `sigdelset`
  - `sigismember`

# Маски сигналов

(у каждого thread'a - своя)

Какие сигналы будут обработаны →

Маска [заблокированных] сигналов →

0	0	0	0	0	1	0
1	0	1	1	1	1	1
SIGHUP	SIGINT	SIGQUIT	SIGILL	SIGABRT	SIGFPE	SIGKILL
0	1	0	0	0	1	0

**Process 1**

`kill(SIGINT, PID)`

**Process 2**

`kill(SIGINT, PID)`

**Process 3**

`kill(SIGFPE, PID)`

*Заблокированные сигналы не исчезают, а "откладываются" до тех пор, пока не будут разблокированы*

Демо: [sigprocmask.c](#)

# Ожидание поступления сигнала

- `pause()` - приостановить выполнение до прихода и обработки **любого незаблокированного** сигнала
- `sigsuspend(sigset_t *set)` - приостановить выполнение до прихода **инвертированного множества** сигналов, **игнорируя все остальные**

# Файловый дескриптор `signalfd`

*Только Linux*

```
int signalfd(int old_signal_fd_or_minus_1,  
             const sigset_t *mask,  
             int flags)
```

- создает файловый дескриптор для "чтения" событий о поступающих сигналах из множества `mask`
- можно читать объекты типа `struct signalfd_siginfo`

POSIX.1b real-time extensions; реализованы в Linux

# **СИГНАЛЫ РЕАЛЬНОГО ВРЕМЕНИ**

# Обычные сигналы

- Имеют стандартное назначение для большинства UNIX-подобных систем
- Процесс может проверить факт того, что во время его паузы ему были доставлены сигналы, но не их количество
- Обработать поступившие сигналы процесс имеет право в произвольном порядке



# Сигналы реального времени

- Имеют номера от `SIGRTMIN` до `SIGRTMAX`
- Могут быть использованы как обычные сигналы (доставка через `kill`)
- Могут быть доставлены через очередь доставки (в этом случае гарантируется порядок доставки и количество)
- Могут нести дополнительную информацию - целое число в поле `si_value` структуры `siginfo_t`

# Отправка сигналов реального времени

## POSIX:

```
sigqueue(int pid,  
         int signum,  
         const union sigval value)
```

```
union sigval {  
    int    sival_int;  
    void*  sival_ptr;  
};
```

## LINUX:

```
sys_rt_sigqueueinfo(  
    int pid,  
    int signum,  
    const siginfo_t  
        *uinfo  
)
```

