Лекция 16

Сигналы

Сигналы (*программные прерывания*) оповещают процессы о событиях:

- аппаратные исключения;
- ввод специальных символов терминала;
- программные события.

Поведение по умолчанию:

- сигнал игнорируется (Ign);
- процесс завершается (Term);
- генерируется *дамп ядра* и процесс завершается (Core);
- процесс приостанавливается (Stop);
- процесс возобновляется (Cont).

Изменение поведения (*диспозиции*) сигнала:

- выполнение по умолчанию;
- сигнал игнорируется;
- выполняется обработчик сигнала.

Имя	Номер	Описание	Действие по умолчанию
SIGABRT	6	Аварийное завершение	Дамп ядра (**CORE**)
SIGBUS	7	Ошибка доступа к памяти	Дамп ядра (**CORE**)
SIGCHLD	17	Дочерний процесс завершен или остановлен	Игнорируется (**IGN**)
SIGSTOP	19	Обязательная остановка	Остановка выполнения процесса (**STOP**)
SIGCONT	18	Продолжить если остановлен	Возобновляет процесс (**CONT**)

SIGTERM	15	Завершить процесс	Завершает процесс (**TERM**)
SIGINT	2	Прерывание с терминала	Завершает процесс (**TERM**)
SIGKILL	9	Обязательное завершение	Завершает процесс (**TERM**)
SIGUSR1	10	Пользовательский сигнал	Завершает процесс (**TERM**)
SIGUSR2	12	Пользовательский сигнал	Завершает процесс (**TERM**)
SIGWINCH	28	Изменён размер окна терминала	Игнорируется (**IGN**)

```c
#include <signal.h>
#include <unistd.h>
#include <stdio.h>

static void  sigHandler(int sig){
 printf("OK!\n");
}

int main(){
  if (signal(SIGINT, sigHandler) == SIG_ERR)
    printf("signal error");

 do{
   printf("Type ^C\n");
   sleep(5);
 }while(1);

}
```

lab17a.c

“Плохой” код.

lab17c.c

```
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <string.h>

sig_atomic_t  s_counter=0;

void  sigHandler(int sig){
 s_counter=sig;
}

int main(int argc, char *argv[]){
 struct sigaction sa[3];
 int i;

 for(i=0;i<2;i++){
     memset(sa+i, 0, sizeof(sa[0]));
     sa[i].sa_handler=&sigHandler;
 }
 sa[2].sa_handler=SIG_IGN;   //или SIG_DFL
```

```
if(sigaction(SIGINT,     &sa[0], NULL)<0) //кроме SIGKILL и SIGSTOP
  printf("sigaction error");
if(sigaction(SIGUSR1, &sa[1], NULL)<0)
  printf("sigaction error");
if(sigaction(SIGTERM, &sa[2], NULL)<0)
  printf("sigaction error");

do{
  printf("Send a signal!\n");
  if(s_counter==SIGINT){
    printf("You sent ctrl C\n");
    s_counter=0;
  }
  if(s_counter==SIGUSR1){
    printf("The SIGUSR1 signal has been sent\n");
    s_counter=0;
  }
  sleep(3);
 }while(1);
}
```

~> man sigaction

```
struct sigaction {
            void     (*sa_handler)(int);
            void     (*sa_sigaction)(int, siginfo_t *, void *);
            sigset_t   sa_mask;
            int        sa_flags;
            void     (*sa_restorer)(void);
        };
```

```
> ./lab17c
Send a signal!
You sent ctrl C
Send a signal!
Send a signal!
The SIGUSR1 signal has been sent

[1]+  Остановлен    ./lab17c
Send a signal!
Send a signal!

[1]+  Убито               ./lab17c
```

```
> ps -aux | grep lab17c
malkov    9028  0.0  0.0   4292   792 pts/3    S+   17:45   0:00 ./lab17c
> kill -s SIGINT  9028
> kill -s SIGUSR1  9028
> kill -s SIGSTOP  9028
> ps -aux | grep lab17c
malkov    9028  0.0  0.0   4292   792 pts/3    T    17:45   0:00 ./lab17c
> kill -s SIGCONT  9028
> kill -s SIGTERM  9028
> kill -s SIGKILL  9028
```

```c
#include <signal.h>
#include <sys/types.h>
#include <stdlib.h>


int main(int argc, char** argv){
        int pid=atoi(argv[1]);
        int idata=atoi(argv[2]);
        int sig=atoi(argv[3]);
        union sigval sv;

       //kill(pid, sig);

        sv.sival_int=idata;
        sigqueue(pid, sig, sv);


 return 0;
}
```

IPC

lab17s.c

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

IPC

lab17r.c

```
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <string.h>

sig_atomic_t  s_counter=0;
sig_atomic_t int_data;


void  sigHandler(int sig){
   s_counter=sig;
}

void sigHandlerA(int sig, siginfo_t* si, void* d){
   s_counter=sig;
   int_data=si->si_value.sival_int;
 }
```

```
int main(int argc, char *argv[]){
 struct sigaction sa[3];
 int i;

 for(i=0;i<2;i++){
     memset(sa+i, 0, sizeof(sa[0]));
 }
 sa[0].sa_handler=&sigHandler;

 sa[1].sa_flags=SA_SIGINFO;
 sa[1].sa_sigaction=&sigHandlerA;

 sa[2].sa_handler=SIG_IGN;   //или SIG_DFL

 if(sigaction(SIGINT, &sa[0], NULL)<0) //кроме SIGKILL и SIGSTOP
    printf("sigaction error");
 if(sigaction(SIGUSR1, &sa[1], NULL)<0)
    printf("sigaction error");
 sigaction(SIGTERM, &sa[2], NULL);
```

```
do{
   printf("Send a signal! %d\n", s_counter);
   if(s_counter==SIGINT){
     printf("You sent ctrl C\n");
     s_counter=0;
   }

   if(s_counter==SIGUSR1){
     printf("data: %d\n", int_data);
     s_counter=0;
   }

   sleep(3);
 }while(1);
}
```

```
siginfo_t {
        int      si_signo;     /* Signal number */
        int      si_errno;     /* An errno value */
        int      si_code;      /* Signal code */
        int      si_trapno;    /* Trap number that caused
                                  hardware-generated signal
                                  (unused on most architectures) */
        pid_t    si_pid;       /* Sending process ID */
        uid_t    si_uid;       /* Real user ID of sending process *
/
        int      si_status;    /* Exit value or signal */
        clock_t  si_utime;     /* User time consumed */
        clock_t  si_stime;     /* System time consumed */
        sigval_t si_value;     /* Signal value */
……………………………………………………………………
};
```

```
> ps -aux | grep lab17r
malkov   7751  0.0  0.0   4292   700
pts/3    S+   16:51   0:00 ./lab17r
> ./lab17s 7751  564 10
> ./lab17s 7751  777 10
> kill -s SIGKILL 7751
```

```
Send a signal!
Send a signal!
data: 564
Send a signal!
……
Send a signal!
data: 777
Send a signal!
……
Send a signal!
Убито
```