Лабораторная работа №7. Работа с сигналами в Unix

Цели работы

1. Получение навыков при работе с сигналами.

Используемое программное обеспечение

При выполнении лабораторной работы будет использовано следующие про-граммное обеспечение: gcc – компилятор C/C++, vi – текстовый редактор.

Порядок выполнения лабораторной работы

1. Войти в систему Linux, указав имя и пароль, предварительно получив их у администратора.
2. В каталоге вашей группы создать файл программу с расширением *.cpp по варианту.
3. Написать программу на тему «Работа с сигналами в Unix».
4. Собрать программу.
5. Исполнить программу

Пример выполнения лабораторной работы

```
/* Демонстрация работы с longjmp/setjmp и сигналами */
#include <iostream>
#include <fcntl.h>
#include <signal.h>
#include <setjmp.h>

using namespace std;
/*#define IGN*/ /* потом откомментируйте эту строку */

jmp_buf cs_stack;   /* control point */
int in_cs;  /* флаг, что мы в критической секции */
int sig_recd;   /* флаг signal received */

/* активная задержка */
void Delay(){
 int i;
 for( i=0; i < 10000; i++ ){
     i += 200; i -= 200;
 }
}

void interrupt( code ){
    cerr<<"\n\n***\n";
    if(code==1){
        cerr<<"*** Обрабатываем сигнал разрешенный";
    }
    else{
        err<<"*** Обрабатываем сигнал отложенный";
    }
    cerr<<"***\n\n" ;
}

/* аргумент реакции на сигнал - номер сигнала (подставляется системой) */
void mexit( nsig ){
    cerr<<"\nУбили сигналом #"<<nsig<<"d...\n\n";
  exit(0);
}

void main(){
    extern void sig_vec(); int code; int killable = 1;

    signal( SIGINT,  mexit );
    signal( SIGQUIT, mexit );
    cerr<<"Данная программа перезапускается по сигналу INTR\n";
    cerr<<"Выход из программы по сигналу QUIT\n\n\n";
```

```
    cerr<<"Сейчас вы еще можете успеть убить эту программу...\n\n";

    Delay(); Delay(); Delay();

    for(;;){
     if( code = setjmp( cs_stack )){
      /* Возвращает не 0, если возврат в эту точку произошел
       * по longjmp( cs_stack, code ); где code != 0
       */
          interrupt( code );/* пришло прерывание */
     } /* else setjmp() возвращает 0,
     * если это УСТАНОВКА контрольной точки (то есть
     * сохранение регистров SP, PC и других в буфер cs_stack),
     * а не прыжок на нее.
     */
      signal( SIGINT, sig_vec ); /* вызывать по прерыванию */
      if( killable ){
        killable = 0;
        cerr<<"\7Теперь сигналы INTR обрабатываются особым образом\n\n\n";
      }
      body(); /* основная программа */
    }
}

void body(){
 static int n = 0; int i;

 cerr<<"\tВошли в тело "<<++n<<"-ый раз\n";
 ecs();
 for( i=0; i < 10 ; i++ ){
  cerr<<"- "<<i<<"\n";
  Delay();
 }
 lcs();
 for( i=0; i < 10 ; i++ ){
  cerr<<"+ "<<i<<"\n";
  Delay();
 }
}

/* запоминание полученных сигналов */
void sig_vec(nsig){
  if( in_cs ){/* we're in critical section */
    #ifdef IGN
    signal( SIGINT, SIG_IGN );  /* игнорировать */
    cerr<<"Дальнейшие прерывания будут игнорироваться\n";
    #else
    signal( SIGINT, sig_vec );
    cerr<<"Дальнейшие прерывания будут подсчитываться\n";
    #endif
    cerr<<"Получен сигнал и отложен\n";
    sig_recd++  ;  /* signal received */
     /* пометить, что сигнал пришел */
  }
  else{
     signal( SIGINT, sig_vec );
     cerr<<"Получен разрешенный сигнал: прыгаем на рестарт\n";
     longjmp( cs_stack, 1);
  }
}

void ecs(){  /* enter critical section */
    cerr<<"Откладываем прерывания\n";
    sig_recd = 0;in_cs = 1;
}

void lcs(){  /* leave critical section */
    cerr<<"Разрешаем прерывания\n";
    in_cs = 0;
    if( sig_recd ){
        cerr<<"Прыгаем на рестарт, т.к. есть отложенный сигнал ("<<sig_recd<<" раз)\n";
```

```
        sig_recd = 0;
        signal( SIGINT, sig_vec );
        longjmp( cs_stack, 2);
    }
}
```

Варианты задания на лабораторную работу

1. Напишите программу, выдающую на экран файл /etc/termcap. Пере-хватывайте сигнал SIGINT, при получении сигнала запрашивайте “Про-должать?”. По ответу ‘y’ - продолжить выдачу; по ‘n’ - завершить програм-му; по ‘r’ - начать выдавать файл с начала:
   lseek(fd,0L,0). Не забудьте заново переустановить реакцию на SIGINT, поскольку после получения сигнала реакция автоматически сбра-сывается. Сигнал прерывания можно игнорировать. Это делается так:

```
signal (SIGINT, SIG_IGN);
```

Такую программу нельзя прервать с клавиатуры. Напомним, что реакция SIG_IGN сохраняется при приходе сигнала.

```
#include <signal.h>
void onintr(sig){                   /* sig - номер сигнала  */
  signal (sig, onintr);            /* восстановить реакцию */
  ... запрос и действия ...
}
main()
{
    signal (SIGINT, onintr);
    ...
}
```

2. Напишите программу, которая ожидает ввода с клавиатуры в тече-ние 10 секунд. Если ничего не введено - печатает Нет «ввода», иначе - печатает «Спасибо». Для ввода можно использовать как вызов read, так и функцию gets (или getchar), поскольку функция эта все равно внутри себя издает системный вызов read. Исследуйте, какое значение возвращает fgets (gets) в случае прерывания ее системным вызовом.
3. Напишите функцию sleep(n), задерживающую выполнение програм-мы на n секунд.
   Воспользуйтесь системным вызовом alarm(n) (будильник) и вызовом pause(), который задерживает программу до получения любого сигнала. Предусмотрите рестарт при получении во время ожидания другого сигна-ла, нежели SIGALRM. Сохраняйте заказ alarm, сделанный до вызова sleep (alarm выдает число секунд, оставшееся до завершения предыдущего зака-за).
4. Напишите «часы», выдающие текущее время каждые 3 секунды.
5. Если завершается процесс, то на экран выводится сообщение: «Завершить процесс?»
6. Если завершается дочерний процесс, то на экран выводится сообщение: «Завершить дочерний процесс?»
7. Напишите “будильник”, выдающие звуковой сигнал через указанное время.
8. Написать программу, завершающую процесс через определенное время.

Требования к отчёту по лабораторной работе

Отчёт должен содержать:

1. Титульный лист
2. Вариант задания
3. Блок-схему и листинг программы.
4. Вывод