Лабораторна робота №14

Тема: Сигнали і таймери в UNIX-подібних операційних системах.

Мета роботи: навчитися програмувати сигнали і таймери UNIX-подібних операційних системах реального часу.

Теоретичні відомості

Для довідок:

```
#define     SIGHUP    1     /* hangup */
#define     SIGINT    2     /* interrupt */
#define     SIGQUIT   3     /* quit */
#define     SIGILL    4     /* illegal instruction (not reset when caught) */
#define     SIGTRAP   5     /* trace trap (not reset when caught) */
#define     SIGABRT   6     /* used by abort, replace SIGIOT in the future */
#define     SIGEMT    7     /* EMT instruction */
#define     SIGFPE    8     /* floating point exception */
#define     SIGKILL   9     /* kill (cannot be caught or ignored) */
#define     SIGBUS    10    /* bus error */
#define     SIGSEGV   11    /* segmentation violation */
#define     SIGFMT    12    /* STACK FORMAT ERROR (not posix) */
#define     SIGPIPE   13    /* write on a pipe with no one to read it */
#define     SIGALRM   14    /* alarm clock */
#define     SIGTERM   15    /* software termination signal from kill */
#define     SIGSTOP   17    /* sendable stop signal not from tty */
#define     SIGTSTP   18    /* stop signal from tty */
#define     SIGCONT   19    /* continue a stopped process */
#define     SIGCHLD   20    /* to parent on child stop or exit */
#define     SIGTTIN   21    /* to readers pgrp upon background tty read */
#define     SIGTTOU   22    /* like TTIN for output if (tp->t_local&LTOSTOP)
*/
#define     SIGUSR1 30      /* user defined signal 1 */
#define     SIGUSR2 31      /* user defined signal 2 */
#define     IGRTMIN   23    /* Realtime signal min */
#define     IGRTMAX   29    /* Realtime signal max */
```

Синтаксис:

sigemptyset() – ініціалізує порожню маску сигналів (всі сигнали виключені) (POSIX)
sigfillset() – ініціалізує повну маску сигналів (всі сигнали включені) (POSIX)
sigaddset() – додає сигнал в маску сигналів (POSIX)
sigdelset() – видаляє сигнал із маски сигналів (POSIX)
sigismember() – перевіряє наявність сигналу в масці сигналів (POSIX)
signal() – вказує обробник сигналу

sigaction() – опитує і/або вказує обробник сигналу (POSIX)
sigprocmask() – опитує і/або змінює маску сигналів (POSIX)
sigpending() – опитує множину заблокованих сигналів, призначених даному процесу (POSIX)
sigsuspend() – призупиняє задачу до отримання сигналу (POSIX)
pause() – призупиняє задачу до отримання сигналу (POSIX)
sigtimedwait() – знімає сигнал з очікування (якщо він не виставлений)
sigwaitinfo() – знімає сигнал з с предельным временем его ожидания
sigvec() – встановлює обробник сигналу
sigsetmask() – встановлює маску сигналів
sigblock() – поповнює маску сигналів
raise() – посилає сигнал викликаній задачі
kill() – посилає сигнал завершення задачі (POSIX)
sigqueue() – посилає сигнал (із черги сигналів) задачі

int sigInit(void)

int sigqueueInit(int nQueues)

int sigemptyset(sigset_t *pSet)

int sigfillset(sigset_t *pSet)

int sigaddset(sigset_t *pSet, int signo)

int sigdelset(sigset_t *pSet, int signo)

int sigismember(const sigset_t *pSet, int signo)

void (*signal (int signo, void (*pHandler) ()))()

int sigaction (int signo, const struct sigaction *pAct, struct sigaction *pOact)

int sigprocmask(int how, const sigset_t *pSet, sigset_t *pOset)

int sigpending(sigset_t *pSet)

int sigsuspend(const sigset_t *pSet)

int pause(void)

int sigtimedwait(const sigset_t *pSet, struct siginfo *pInfo, const struct timespec *pTimeout)

int sigwaitinfo(const sigset_t *pSet, struct siginfo *pInfo)

int sigvec(int sig, const struct sigvec *pVec, struct sigvec *pOvec)

int sigsetmask(int mask)

int sigblock(int mask)

int raise(int signo)

int kill(int tid, int signo)

int sigqueue(int tid, int signo, const union sigval value)

2. Завдання

Написати програму, яка:

1. Виводить текст привітання при отриманні сигналу SIGALARM.
2. Виводить ім’я автора при отриманні SIGUSR1.
3. Запускає процес при отриманні сигналу SIGUSR2.
4. Завершує і виводить повідомлення про свою роботи при отриманні сигналу 29.

Приклад:

Програма, яка посилає сигнал

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>
//using namespace std;

int main( int argc, char** argv)
{
	int pid, sig = SIGTERM;
	if (argc==3)
	{
		if (sscanf(argv[1],"%d",&sig)!=1)
		{	/* get signal number */
			//cerr << "Invalid signal: " << argv[1] << endl;
			printf ("Invalid signal: %d \n", argv[1]);
			return -1;
		}
		argv++, argc--;
	}
	while (--argc > 0)
		if (sscanf(*++argv,"%d",&pid)==1)
		{	/* get process ID */
			if (kill (pid, sig)==-1)
			perror("kill");
		}
		else //cerr << "Invalid pid: " << argv[0] << endl;
	printf ("Invalid pid: %d \n", argv[0]);
	return 0;
}
```

Програма, яка приймає сигнал

```
#include <stdio.h>
```

```
#include <unistd.h>
#include <signal.h>
#define   INTERVAL   50
void callme( int sig_no )
{
        //alarm( INTERVAL );
        //printf("PREVED!!! %d\n", sig_no);
        switch (sig_no)
        {
                case 14: printf("PREVED!!!\n");
                        break;
                case 10: printf("Author\n");
                        break;
                case 12: printf("Start process\n");
                        execl("/bin/sh", "sh", "-c", "dir", 0);
                        break;
                case 29: printf("Bye Bye!!!\n");
                        exit(0);
                        break;
        }
}

int main()
{
        sigset_t  sigmask;
        struct sigaction  action;
        sigemptyset(&action.sa_mask);
        action.sa_handler = (void (*)())callme;
        action.sa_flags = 0;
        if ( sigaction( SIGUSR1,&action,0 )==-1 )
        {
                perror( "sigaction");
                //return 1;
        }
        if (sigaction(SIGUSR2,&action,0)==-1)
                perror( "sigaction");
        if (sigaction(SIGIO,&action,0)==-1)
                perror( "sigaction");
        if (sigaction(SIGALRM,&action,0)==-1)
                perror( "sigaction");
        printf("My pid is %i\n", getpid());
        printf("Waiting...\n");
        while (1)
                pause();          /* wait for signal interruption */
        return 0;
}
```

Написати програму, в якій:

1. Встановити поточний час 22:43:30, 20 січня 2012.
2. Вивести дату і поточний час.
3. Визначити (в мксек.) і порівняти час обчислення функцій sin(x) і tg(x).

4. Визначити (в мксек.) і час обчислення функції sin(x) використовуючи таймер.
5. Вивести ще раз поточний час.

Приклад:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
#include <unistd.h>

struct timeval tv1,tv2;
struct tm tmm;
struct timespec ttmm;

void calc_sin(int sig, siginfo_t *siginf, void *ptr)
{
        int i, ntimes=100000;
        float d1;
        gettimeofday(&tv1, NULL);
        for(i=0;i<ntimes;i++)
        d1=sin(i);
        gettimeofday(&tv2, NULL);
        tv2.tv_usec += 1000000*(tv2.tv_sec-tv1.tv_sec);
        d1 = (float)(tv2.tv_usec - tv1.tv_usec)/1000000.;
        printf("\ndelta sec for SIN %f \n", d1);
}

int calc_tg()
{
        int i, ntimes=100000;
        float d1;
        gettimeofday(&tv1, NULL);
        for (i=0;i<ntimes;i++)
        d1=tan(i);
        gettimeofday(&tv2, NULL);
        tv2.tv_usec += 1000000*(tv2.tv_sec-tv1.tv_sec);
        d1 = (float)(tv2.tv_usec - tv1.tv_usec)/1000000.;
        printf("\ndelta sec for TG %f \n", d1);
}

int set_time()
{
        tmm.tm_year = 2012 - 1900;
        tmm.tm_mon = 0;
        tmm.tm_mday = 20;
        tmm.tm_hour = 21;
        tmm.tm_min = 43;
        tmm.tm_sec = 30;
        tmm.tm_isdst = -1;
```

```
	ttmm.tv_sec = mktime(&tmm);
	clock_settime(CLOCK_REALTIME, &ttmm);
	printf("set time\n");
}

int get_time()
{
	clock_gettime(CLOCK_REALTIME, &ttmm);
	gmtime_r(&ttmm.tv_sec, &tmm);
	printf("NOW = %s\n",asctime(&tmm) );
}

void callme( int signo, siginfo_t* evp, void* ucontext )
{
	time_t tim = time(0);
	printf ("callme: %s, signo: %s, %s", evp->si_value.sival_int, signo, ctime(&tim));
}

int mktimer()
{
	struct sigaction   sigv;
	struct sigevent    sigx;
	struct itimerspec  val;
	struct tm  do_time;
	timer_t  t_id;
	//signal(SIGUSR1, callme);
	sigemptyset(&sigv.sa_mask);
	sigv.sa_flags = SA_SIGINFO;
	sigv.sa_sigaction = calc_sin;
	//sigv.sa_sigaction = callme;
	if (sigaction( SIGUSR1, &sigv, 0) == -1)
	{
		perror("sigaction");
		return 1;
	}
	sigx.sigev_notify = SIGEV_SIGNAL;
	sigx.sigev_signo  = SIGUSR1;
	sigx.sigev_value.sival_int = 12;
	if ( timer_create( CLOCK_REALTIME, &sigx, &t_id ) == -1)
	{
		perror("timer_create");
		return 1;
	}
	/* Set timer to go off at April 20, 1996, 10:27am */
	do_time.tm_hour = 10;
	do_time.tm_min = 3;
	do_time.tm_sec = 55;
	do_time.tm_mon = 11;
	do_time.tm_year = 107;
	do_time.tm_mday = 7;
	val.it_value.tv_sec = mktime( &do_time );
	val.it_value.tv_nsec = 0;
```

```
        val.it_interval.tv_sec = 5;
        val.it_interval.tv_nsec = 0;
        //cerr << "timer will go off at: " << ctime(&val.it_value.tv_sec);
        printf ("timer will go off at: %s", ctime(&val.it_value.tv_sec));
        if (timer_settime( t_id, TIMER_ABSTIME, &val, NULL ) == -1 )
        {
                perror("timer_settime");
                return 2;
        }
        /* do something then wait for the timer to expire twice*/
        int i;
        for (i=0; i < 2; i++ )
                pause();
        if (timer_delete( t_id ) ==-1)
        {
                perror( "timer_delete" );
                return 3;
        }
        return 0;
}

int main(int argc, char **argv)
{
        char wait[1];
        printf("program started\n");
        scanf("%s",wait);
        set_time();
        scanf("%s",wait);
        get_time();
        scanf("%s",wait);
        calc_sin();
        scanf("%s",wait);
        calc_tg();
        scanf("%s",wait);
        mktimer();
        scanf("%s",wait);
        get_time();
}
```

3. Зміст звіту

1. Титульна сторінка та тема роботи.
2. Мета роботи.
3. № варіанту та завдання.
4. Текст програм і протокол їх роботи в системі.
5. Висновки.

4. Контрольні запитання

1. Назвіть способи міжзадачного обміну.
2. Суть поняття “повідомлення” (*message*) в багатозадачних ОС РЧ.
3. Суть терміну “*mail box”* в міжзадачному обміні.

4. Що таке “сигнал” (*signal*), його суть в ОС UNIX.
5. Як створити обробник сигналу в ОС UNIX.