МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
(НИЯУ МИФИ)
Институт интеллектуальных кибернетических систем
Кафедра Кибернетики

Лабораторная работа №3

Выполнила студентка группы Б17-501: Баранова Дарья

Проверил: Ктитров С.В

Москва, 2020

Задача

Вариант С–2

Тема «Таймеры POSIX»

Разработать программу. Программа-родитель запускает заданное число процессов, рассчитывающих значения элементарных функций от матриц через разложение в ряд, а затем по истечении заданного для каждого процесса времени проверяет, закончен ли процессом расчет. Использовать возможность передачи информации вместе с сигналом. Программа должна собираться из нескольких файлов с использованием make.

Выполнение

В командной строке передаются целочисленные аргументы – по их количеству запускаем столько же процессов. Для каждого процесса создается отдельный таймер.

Таймер устанавливается на время, которое было получено из командной строки.

Когда истекает время таймера, он срабатывает и отправляет сигнал обработчику. Обработчик проверяет его – если сигнал послан таймером, то выводит сообщение “Вычисление не окончено”, иначе “Вычисление окончено”.

В качестве функции для вычисления в каждом процессе была выбрана функция вычисления экспоненты от матрицы размера 300*300.

Код программы:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <errno.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <signal.h>
#include <sys/signal.h>

int matrixSize = 300;
void hdl (int sig, siginfo_t *siginfo, void *context) {
    int int_val = siginfo->si_value.sival_int;
    if (int_val != 1){
        printf("Вычисление не окончено. pid: (%d)\n", int_val);
    } else {
        printf("Вычисление завершено! pid: (%d)\n", siginfo->si_pid);
    }
}

void MatrixFunction() {
    int n = matrixSize;
    double a[n * n];
    for (int i = 0; i < n * n; i++) {
        a[i] = i + 1;
    }
    double EPS = 0.1;
    int i, j, k, itr;
    double am, em, curEPS;
    double matrix_C[n * n];
    double matrix_B[n][n];
    double answer[n * n];
    em = 0.;
```

```
36      for (i = 0; i < n; i++) {
37          for (j = 0; j < n; j++) {
38              answer[i * n + j] = 0.;
39              matrix_B[i][j] = 0.;
40              am = a[i * n + j];
41              if (am < 0) {
42                  am = -am;
43              }
44              if (am > em) {
45                  em = am;
46              }
47
48          }
49          answer[i * n + i] = 1.;
50          matrix_B[i][i] = 1.;
51      }
52      curEPS = 1.;
53      itr = 0;
54      while (curEPS > EPS) {
55          itr++;
56          if (itr >= 40) break;
57          curEPS = 0.;
58          for (j = 0; j < n; j++){
59              for (i = 0; i < n; i++){
60                      matrix_C[i] = matrix_B[i][j];
61                  }
62              for (i = 0; i < n; i++) {
63                  matrix_B[i][j] = 0.;
64                  for (k = 0; k < n; k++) {
65                      matrix_B[i][j] += a[i * n + k] * matrix_C[k];
66                  }
67              }
68          }
```

```
69          for (i = 0; i < n; i++) {
70              for (j = 0; j < n; j++) {
71                  matrix_B[i][j] /= (double)itr;
72                  answer[i * n + j] += matrix_B[i][j];
73                  am = abs(matrix_B[i][j]);
74                  if (am > curEPS) {
75                      curEPS = am;
76                  }
77              }
78          }
79      }
80  }
81
82  int makeTimer(timer_t *timerID, int expireMS, int intervalMS) {
83      struct sigevent te;
84      struct itimerspec its;
85      struct sigaction sa;
86      int sigNo = SIGRTMIN;
87
88      /* Set up signal handler. */
89      sa.sa_flags = SA_SIGINFO;
90      sa.sa_sigaction = hdl;
91      sigemptyset(&sa.sa_mask);
92      if (sigaction(sigNo, &sa, NULL) == -1) {
93          perror("sigaction");
94      }
95
96      /* Set and enable alarm */
97      te.sigev_notify = SIGEV_SIGNAL;
98      te.sigev_signo = sigNo;
99      te.sigev_value.sival_int = getpid();
100     timer_create(CLOCK_REALTIME, &te, timerID);
101     memset(&its, 0, sizeof (its));
```

```
102     its.it_interval.tv_sec = 0;
103     its.it_interval.tv_nsec = 0;
104     its.it_value.tv_sec = expireMS;
105     its.it_value.tv_nsec = 0;
106     timer_settime(*timerID, 0, &its, NULL);
107
108     return 1;
109 }
110
111 void ProcessWork(pid_t PID, int sleep_time, timer_t timerID) {
112     makeTimer(&timerID, sleep_time, 0);
113     MatrixFunction();
114     union sigval value;
115     value.sival_int = 1;
116     sigqueue(PID, SIGRTMIN, value);
117 }
118
119 int main(int argc, char* argv[]) {
120     int n = argc - 1;
121     int times[n];
122     for (int i = 1; i < argc; i++) {
123         times[i-1] = atoi(argv[i]);
124     }
125     pid_t pid = 0;
126     int PID = getpid();
127     timer_t timerID[n];
128
129     pid_t all_pid[n];
130     struct sigaction act;
131     memset (&act, '\0', sizeof(act));
132     act.sa_sigaction = &hdl;
133     act.sa_flags = SA_SIGINFO;
134     sigaction(SIGRTMIN, &act, NULL);
```

```
135
136     for (int i = 0; i < n; i++) {
137         pid = fork();
138         if (pid != 0) {
139             ProcessWork(pid, times[i], timerID[i]);
140             break;
141         }
142     }
143     for (int i = 0; i < n; i++) {
144         wait(&all_pid[i]);
145     }
146 }
147
```

Код Makefile:

```
SHELL = /bin/bash

run:
    cc main.c -o main -lrt
    ./main 3 7 5 1 1
```

1 урок!

Пример работы:

```
super@super-Inspiron-5558:~/Desktop/unik/C$ cc main.c -o main -lrt
super@super-Inspiron-5558:~/Desktop/unik/C$ ./main 1 2 3
Вычисление не окончено. pid: (1088)
Вычисление не окончено. pid: (1089)
Вычисление завершено! pid: (1090)
Вычисление завершено! pid: (1088)
Вычисление завершено! pid: (1089)
```

```
super@super-Inspiron-5558:~/Desktop/unik/C$ make
cc main.c -o main -lrt
./main 5 6 2 4 8 1 3
Вычисление не окончено. pid: (4750)
Вычисление не окончено. pid: (4747)
Вычисление завершено! pid: (4751)
Вычисление завершено! pid: (4748)
Вычисление завершено! pid: (4745)
Вычисление завершено! pid: (4746)
Вычисление завершено! pid: (4749)
Вычисление завершено! pid: (4747)
Вычисление завершено! pid: (4750)
```