ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

ФАКУЛЬТЕТ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Кафедра вычислительных систем

ОТЧЁТ
по лабораторной работе №3
по дисциплине «Распределённые системы и технологии»

Выполнил:
студент группы МГ-165
Терешков Р. В.

Проверил:
ст. пр. Фульман В. О.

Новосибирск – 2016

Задание на проектирование

1. Разработать программу, реализующую модель работы склада, отвечающего за хранение и продажу некоторого товара (одного). Склад содержит N помещений, каждый из которых может хранить определённое количество единиц товара. Поступающий товар помещается в одно из помещений специальным погрузчиком. За товаром прибило K покупателей, каждому из которых требуется по Lk единиц товара. Площадка перед складом мала и на ней может в один момент времени находиться либо погрузчик, либо один из покупателей. Если покупателям требуется больше товара, чем имеется на складе, то они ждут новых поступлений, периодически проверяя склад. Время работы склада ограниченно.

- Основная нить (функция main) выполняет следующие действия:
    - Формирует начальное заполнение склада (для каждого помещения случайным образом выбирается число из диапазона от 1 до 40);
    - Обрабатывает опции командной строки, в которой должно быть указано сколько клиентов будет обслуживаться складом и в течении какого времени должен склад работать;
    - Порождает заданное количество нитей, каждая из которых реализует алгоритм работы покупателя. Каждому покупателю случайным образом назначается количество требуемых единиц продукции (число из диапазона от 1 до 1000).
    - Настраивает таймер (alarm) таким образом, чтобы он сработал по окончании времени работы склада;
    - Запускает алгоритм работы погрузчика;
    - После срабатывания таймера принудительно завершает все выполняющиеся нити (если таковые имеются).
    - Завершает работу программы.
- Алгоритм работы погрузчика.
    - Пытается попасть на площадку перед складом;
    - Как только попадает на площадку, ищет хотя бы один склад, в котором нет продукции, и заполняет его максимально возможным образом;
    - Покидает площадку;
    - «засыпает» на 5 секунд;
    - Цикл повторяется до срабатывания таймера;
- Алгоритм работы покупателя.
    - Пытается попасть на площадку перед складом;
    - Как только попадает на площадку, ищет хотя бы один склад, в котором есть продукция, и забирает либо столько, сколько надо, либо всю продукцию;
    - Покидает площадку;
    - «засыпает» на 5 секунд;
    - Цикл повторяется до тех пор, пока покупателю нужна продукция;

Программа должна на экран выводить информацию о помещениях склада.

2. Доработайте программу умножения матриц из лабораторной работы № 2 с наилучшим способом обхода оперативной памяти так, чтобы использовалось автоматическое распараллеливание циклов for. Продемонстрируйте, что результат умножения матриц получился правильным. Оцените получившееся ускорение выполнения программы.

3. Спроектируйте и разработайте параллельное приложение, реализующее игру «крестики нолики».
Одна нить отвечает за интерактивное взаимодействие с пользователем. Ожидает ввод с клавиатуры определённых клавиш (остальные клавиши игнорируются). Если пользователь нажимает клавишу S, то нить сообщает второй нити, что можно начать или прекратить «играть» (см. ниже). Если нажата клавиша T, то пользователю предлагается ввести целое число в диапазоне от 1 до 10. После ввода первая нить меняет значение общей переменной timeThink. Если пользователь нажимает клавишу A, то ему предлагается ввести целое число в диапазоне от 5 до 30, которое записывается в общую переменную timeRestart.
Вторая нить реализует имитацию игры «Крестики-нолики» между двумя игроками. Каждый игрок «думает» в течение времени timeThink и делает свой ход случайным образом в любую свободную ячейку. Игра продолжается до тех пор, пока какой-то из игроков не выиграет или не будет заполнено все поле. После окончания игры происходит перезапуск после ожидания timeRestart секунд.

Ход выполнения работы

1. Данное приложение разработано с использованием потоков из стандарта POSIX Threads. Программа моделирует работу склада, содержащего некоторое число помещений с товарами. Алгоритм подразумевает параллельное выполнение функций погрузчика, покупателя и отображения текущего состояния склада. На вход программе подаются три аргумента командной строки: количество помещений склада, количество покупателей и время работы. Для определения времени работы используется таймер, срабатывающий по сигналу `SIGUSR1`. Пример работы программы:

```
[tereshkov@jet task1]$ ./a.out 60 2 50
Hello there. Your input parameters:
Quantity of rooms in the store: 60
Quantity of customers: 2
Uptime: 50 secs.
Initial rooms state:
  30    2   32   35   38   27   33   20    4    9
   9   14    9   30   18   35   19   11    3    5
  29   28   21   26   18   15    5    2   39   15
  27   29   16   10   23   13   28    7   25   24
  16   33   37   24   14    6   10   32   16    5
  28   36   24    1   13    2   15   17    3   14
Customers stuff:
 751  942
Hi, I'm customer 0!
I`ll (0) take that from 0!
Hi, I'm customer 1!
I`ll (1) take that from 24!
Hi, I'm dozer!
I`ll fill case: 0 w/ 40 things!
Rooms state:
  40    2   32   35   38   27   33   20    4    9
   9   14    9   30   18   35   19   11    3    5
  29   28   21   26    0   15    5    2   39   15
  27   29   16   10   23   13   28    7   25   24
  16   33   37   24   14    6   10   32   16    5
  28   36   24    1   13    2   15   17    3   14
I`ll (1) take that from 41!
I`ll (0) take that from 34!
I`ll fill case: 24 w/ 40 things!
Rooms state:
  40    2   32   35   38   27   33   20    4    9
   9   14    9   30   18   35   19   11    3    5
  29   28   21   26   40   15    5    2   39   15
  27   29   16   10    0   13   28    7   25   24
  16    0   37   24   14    6   10   32   16    5
  28   36   24    1   13    2   15   17    3   14
^C
```

Инициализация и установка таймера осуществляется с помощью следующих функций:

```
timer_t create_timer(int signo)
{
    timer_t timerid;
    struct sigevent se;
    se.sigev_notify = SIGEV_SIGNAL;
    se.sigev_signo = signo;
    timer_create(CLOCK_REALTIME, &se, &timerid);
    return timerid;
}

void set_timer(timer_t timerid, int secs)
{
    struct itimerspec timervals;
    timervals.it_value.tv_sec = secs;
    timervals.it_value.tv_nsec = 0;
    timervals.it_interval.tv_sec = 0;
    timervals.it_interval.tv_nsec = 0;
    timer_settime(timerid, 0, &timervals, NULL);
}

void install_sighandler(int signo, void(*handler)(int))
{
    sigset_t set;
    struct sigaction act;

    act.sa_handler = handler;
    act.sa_flags = SA_RESTART;
    sigaction(signo, &act, 0);

    sigemptyset(&set);
    sigaddset(&set, signo);
    sigprocmask(SIG_UNBLOCK, &set, NULL);
}

void signal_handler(int signo)
{
    if (signo == SIGUSR1)
         work = 0;
}
```

Полный исходный код программы доступен по ссылке.

2. Для автоматического распараллеливания циклов for в программу были добавлены специальные директивы из стандарта OpenMP – `pragma omp parallel` и `pragma omp for`. В доработанной программе также использовался наилучший способ обхода оперативной памяти (перемножение матриц строка на строку):

```
#pragma omp parallel shared(a, b, c) private(q, j)
{
    #pragma omp for
    for (i = 0; i < m; ++i) {
        for (q = 0; q < n; ++q) {
            for (j = 0; j < k; ++j) {
                c[i][j] += a[i][q] * b[q][j];
            }
        }
    }
}
```

Полученные ускорения параллельной версии относительно последовательной при размере матрицы 1000x1000:

OMP_NUM_THREADS	**2**	**4**	**8**
SPEEDUP	1.81	3.32	6.54

Полная версия программы доступна по следующей ссылке.

3. В заключительном задании лабораторной работы было реализовано параллельно приложение, реализующую игру “Крестики-Нолики”. Во время игры пользователь может задавать различные параметры, отвечающие за время на ход, время до начала следующей игры, а также реализован выход из игры по нажатию клавиши ‘q’. Интерактивный режим работы обеспечивается за счёт перевода терминала в неканонический режим:

```
struct termios rk_currentTermState;

int mytermsave()
{
    if (tcgetattr(1, &rk_currentTermState) == -1)
        return -1;
    return 0;
}

int mytermrestore()
{
    if (tcsetattr(1, TCSANOW, &rk_currentTermState) == -1)
        return -1;
    return 0;
}

int mytermregime(int regime, int vtime, int vmin, int echo,
    int sigint)
{
```

```
    struct termios term;

    if ((regime > 1) || (regime < 0) || (echo > 1) ||
        (echo < 0) || (sigint > 1) || (sigint < 0))
        return 1;

    if ((vtime < 0) || (vmin < 0)) return -1;
    if (tcgetattr(1, &term) == -1) return -1;

    if (regime == 0) term.c_lflag &= (~ICANON);
    else term.c_lflag |= ICANON;

    if (echo == 0) term.c_lflag &= (~ECHO);
    else term.c_lflag |= ECHO;

    if (sigint == 0) term.c_lflag &= (~ISIG);
    else term.c_lflag |= ISIG;

    term.c_cc[VMIN] = vmin;
    term.c_cc[VTIME] = vtime;

    if (tcsetattr(1, TCSANOW, &term) == -1) return -1;

    return 0;
}
```

а также изменения параметров таймеров, срабатывающих по сигналам `SIGUSR1` и `SIGUSR2` соответственно. Пример работы программы проиллюстрирован на рисунке снизу.

Полный исходный код доступен по ссылке.

```
***** turn: 1
-|x|-
-|-|-
-|-|-

***** turn: 2
-|x|-
-|-|-
-|o|-

***** turn: 3
x|x|-
-|-|-
-|o|-

***** turn: 4
x|x|-
-|-|o
-|o|-

***** turn: 5
x|x|-
x|-|o
-|o|-
Input a new think time.
Think time is now 1 secs.

***** turn: 6
x|x|-
x|-|o
o|o|-

***** turn: 7
x|x|-
x|x|o
o|o|-

***** turn: 8
x|x|-
x|x|o
o|o|o
Result: `o` wins the game.
```