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 №4
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. Продемонстрировать запуск фонового процесса в системах GNU\Linux.

2. Разработать приложение, порождающее несколько процессов и выводящих информацию о каждом из них. В каждом процессе должны быть выведены значения идентификаторы: PID, PPID, GID, EGID, UID, EUID и т.п.

3. Подготовить приложение, реализующее программу, представленную на рисунке 12. Разработать описание задачи, которой требуется для выполнения 3 вычислительных ядра и необходим запуск созданной программы. Запустите задачу на выполнение. Убедитесь, что приложение выдало правильные значения. Доработать программу так, чтобы выводилась информация о каждом процессе в MPI приложении (информация аналогична пункту 2 простого задания).

4. Разработайте программу умножения матриц с использованием библиотеки MPI. Продемонстрируйте, что результат умножения матриц получился правильным. Оцените получившееся ускорение выполнения программы.

5. Разработайте гибридное приложение (MPI+OpenMPI), реализующее алгоритм умножения матриц.

Ход выполнения работы

1. Для запуска фонового процесса в GNU\Linux необходимо запустить программу с символом `&` или же запустить в обычном режиме, в дальнейшем отправив его на выполнение в фоновом режиме командой `bg`. Для обратного действия используется команда `fg`.

```
[tereshkov@jet lab4]$ ./prog &
[1] 15498
[tereshkov@jet lab4]$ ps aux | grep 15498
tereshk+ 15498  116  0.0   4144   324 pts/0    R    13:48   0:04 ./prog
tereshk+ 15500  0.0  0.0 112656   916 pts/0    S+   13:48   0:00 grep --
[tereshkov@jet lab4]$
```

2. Каждый процесс в GNU\Linux имеет набор идентификаторов, для вывода которых используются системные вызовы. К ним относятся:

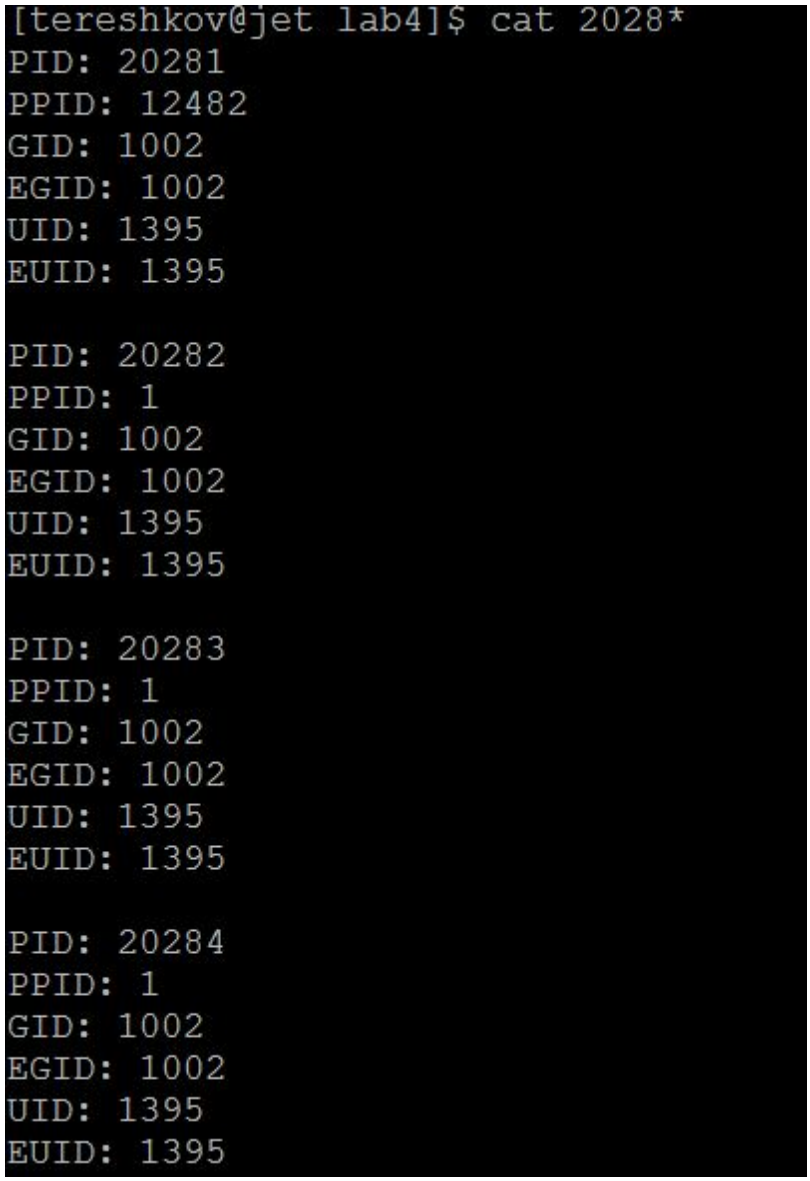
- `pid_t getpid()` — идентификатор процесса;
- `pid_t getppid()` — идентификатор родительского процесса;
- `uid_t getuid()` — идентификатор пользователя;
- `uid_t geteuid()` — «эффективный» идентификатор пользователя;
- `gid_t getgid()` — идентификатор группы пользователя;
- `pid_t getsid(pid_t pid)` — идентификатор сеанса для процесса.

и прочие идентификаторы. В рамках данной работы было реализовано приложение, выводящее для создаваемых процессов соответствующие наборы.

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

int main()
{
    fork();
    fork();
    char filename[64] = "";
    snprintf(filename, sizeof(filename), "%d.pid",(int)getpid());

    FILE *fp = fopen(filename, "w");
    fprintf(fp, "PID: %d\n", getpid());
    fprintf(fp, "PPID: %d\n", getppid());
    fprintf(fp, "GID: %d\n", getgid());
    fprintf(fp, "EGID: %d\n", getegid());
    fprintf(fp, "UID: %d\n", getuid());
    fprintf(fp, "EUID: %d\n\n", geteuid());
    fclose(fp);
    return 0;
}
```



3. Далее по заданию требуется ознакомиться с базовыми аспектами стандарта MPI (Message Passing Interface). Приложения, написанные на языке программирования C с использованием функций одной из реализаций MPI, запускаются с помощью директивы `mpirun` или `mpiexec`. Тестовая программа, доступная по ссылке, производит инициализацию среды MPI, выводит информацию о пространстве обмена информации в приложении, после чего завершает работу. Далее реализованное приложение было усовершенствовано следующим образом: для каждого процесса в MPI в файл выводится набор идентификаторов, соответствующих этому процессу.

```
[tereshkov@jet lab4]$ cat 2320*
Hi, Jack! From processor cn9, rank 0 out of 3 processors.
PID: 23202
PPID: 23201
GID: 1002
EGID: 1002
UID: 1395
EUID: 1395

Hi, Jack! From processor cn9, rank 1 out of 3 processors.
PID: 23203
PPID: 23201
GID: 1002
EGID: 1002
UID: 1395
EUID: 1395

Hi, Jack! From processor cn9, rank 2 out of 3 processors.
PID: 23204
PPID: 23201
GID: 1002
EGID: 1002
UID: 1395
EUID: 1395
```

4. Библиотека MPI позволяет создавать высокоэффективные параллельные приложения, функционирующие за счёт интерфейса обмена сообщениями между процессами. В качестве демонстрации возможности стандарта было реализовано программное обеспечение, выполняющее умножение матриц с использованием библиотеки MPI. Исходный код программы доступен по следующей ссылке.

5. Для эффективного использования топологии некоторых классов вычислительных систем (SMP/NUMA-системы) могут использоваться гибридные приложения, использующие как функции библиотеки MPI, так и директивы OpenMP. В рамках выполнения данной работы было реализовано гибридное приложение (MPI+OpenMP), реализующее алгоритм умножения прямоугольных матриц. Для постановки гибридного приложения на выполнение на вычислительном кластере Jet была описана задача с указанием требуемого значения переменной среды окружения `OMP_NUM_THREADS`. Код программы доступен по ссылке. Время выполнения программ при размере матрицы 1000x1000 отображено в таблице ниже.

N_P (OMP_NUM_THR)	2 (2)	4 (4)	8 (8)
MM_MPI	7.192	2.427	1.094
MM_MPI+OMP	3.528	1.106	1.115