Министерство образования и науки РФ

Севастопольский государственный университет

Отчёт

по лабораторной №7

«Исследование подсистемы управления процессами и потоками в ОС семейства UNIX»

Выполнил:

ст.гр.ИСб-22д

Воронин И.Ю.

Проверил:

Забаштанский А.К.

Севастополь

2015

1.Цель работы

Изучение понятий процесса и потока ОС семейства UNIX. Приобретение практических навыков разработки программ на языке Cи в ОС UNIX, а также написания программ с использованием системных вызовов создания и управления процессами и потоками в ОС UNIX.

.

2.Вариант задания

Написать на языке C программу **Sort**, реализующую следующие действия:

Получить целое число(номер процесса) из первого аргумента программной строки;

Получить с использованием системных вызовов getppid() и getpid() идентификаторы родительского и текущего процесса;

Вывести на экран номер процесса, идентификаторы родительского и текущего процесса;

Заполнить массив целых чисел случайными значениями из диапазона 0-100;

Отсортировать массив;

Вывести на экран отсортированный массив (перед каждым элементом массива выводить номер процесса)

Метод и направление сортировки, а также количество элементов массива N выбирается в соответствии с вариантом задания, приведенным в таблице 3.1.

Написать на языке C программу Master, выполняющую следующие действия:

Получить с использованием системных вызовов getppid() и getpid() идентификаторы родительского и текущего процесса и вывести их на экран;

Используя системные вызовы *fork()* и *exec()* создать M процессов (функция для выполнения системного вызова *exec()* и количество процессов Mопределяются по варианту задания таблица 3.1)

Написать на языке C программу **Threads,** содержащую процедуру сортировки массива - **sort**, и процедуру вывода массива на экран – **mass\_print**. Процедура **sort** должна получать идентификатор собственного потока и выводить его на экран (в формате **sort:pthreadId=***threadId*), после чего сортировать массив. Процедура **mass\_print** должна получать идентификатор собственного потока, выводить его на экран (в формате **mass\_print:pthreadId=***threadId*) и выводить на экран массив. Программа **Threads** должна выполнять следующие действия:

Заполнить массив случайными числами;

Используя системные вызовы *pthread\_create()***,** создать потоки **sort** и **mass\_print** и получить идентификаторы созданных потоков.

Вывести на экран идентификаторы созданных потоков (в формате **threads:sort:pthreadId=***threadId***; mass\_print:pthreadId=***threadId*).

Ожидать завершения потоков (с использованием вызова *pthread\_join*).

Модифицировать программу **Threads** так, чтобы обеспечить синхронизацию потоков. Поток **mass\_print**, после запуска должен ожидать завершения потока **sort** (с использованием вызова *pthread\_join)*, и лишь затем выводить массив на экран (для этого необходимо при создании потока **mass\_print** в параметре *arg* передать идентификатор потока **sort**).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № варианта | Метод сортировки | Направление | N | Функция для выполнения *exec()* | Количество процессов М |
| 1 | Шелла | Убыв. | 30 | *execlp()* | 4 |
| 2 | Быстрая | Убыв. | 35 | *execvp()* | 3 |
| 3 | Пузырька | Убыв. | 40 | *execl()* | 2 |
| 4 | Выбора | Убыв. | 45 | *execv()* | 4 |

Таблица 2.1 - Варианты заданий.

3.Код программы

Main:

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

int main(int argc, const char \*argv[]) {

pid\_t pid, ppid;

pid = getpid ();

ppid = getppid ();

printf ("Pid: %d \n", pid);

printf ("Ppid: %d \n", ppid);

if (!fork ()) {

execve( "Os2", (char\*\*) argv , NULL);

return 1;

}

return 0;

}

Sort:

#include <sys/types.h>

#include <unistd.h>

#include <algorithm>

#include <stdlib.h>

#include <stdio.h>

const int N = 35;

char\* id;

void qs(int\* s\_arr, int first, int last);

void mass\_print(int \*arr, int size);

int main(int argc, char \*argv[]) {

pid\_t pid, ppid;

int arr[N];

if(argc>1){

id = argv[1];

}

pid = getpid();

ppid = getppid();

printf("Pid %d \n", pid);

printf("Ppid %d \n", ppid);

printf("ID %s \n", id);

for(int i=0;i<N;i++){

arr[i]= rand() %100;

}

qs(arr, 0, N);

mass\_print(arr, N);

return 0;

}

void qs(int\* s\_arr, int first, int last){

int i = first, j = last, x = s\_arr[(first + last) / 2];

do {

while (s\_arr[i] > x) i++;

while (s\_arr[j] < x) j--;

if (i <= j) {

if (s\_arr[i] < s\_arr[j]) std::swap(s\_arr[i], s\_arr[j]);

i++;

j--;

}

} while (i <= j);

if (i < last)

qs(s\_arr, i, last);

if (first < j)

qs(s\_arr, first, j);

}

void mass\_print(int \*arr, int size) {

for(int i = 0; i < size; i++) {

printf("ID %s : %d \n", id, arr[i]);

}

}

Threads:

#include <sys/types.h>

#include <algorithm>

#include <unistd.h>

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

const int N = 35;

int arr[N];

int first = 0;

int last = N;

void \*qs(void \*arg);

void \*mass\_print(void \*arg);

int main(){

pid\_t pid = getpid();

printf("Pid %d \n", pid);

int id1 = 100;

int id2 = 102;

pthread\_t thread1;

pthread\_t thread2;

for(int i = 0; i < N;i++){

arr[i]= rand() %100;

}

for(int i = 0; i < N; i++) {

printf(" %d ", arr[i]);

}

printf("\n \n ");

int result1 = pthread\_create(&thread1, NULL, qs ,NULL );

pthread\_join(thread1 , NULL );

int result2 = pthread\_create(&thread2, NULL, mass\_print ,NULL );

pthread\_join(thread2 , NULL );

return 0;

}

void \*qs(void \*arg){

for (int i = 0; i < N-1; i++) {

for (int j = 0; j < N-i-1; j++) {

if (arr[j] > arr[j+1]) {

int b = arr[j]; //change for elements

arr[j] = arr[j+1];

arr[j+1] = b;

}

}

}

return NULL;

}

void \*mass\_print(void \*arg) {

for(int i = 0; i < N; i++) {

printf("ID : %d \n", arr[i]);

}

return NULL;

}

4.Ход работы

В ходе данной работы составляем три файла с кодом программы на языке С++. Компилируем их при помощи команды g++ в окне bash.

Запуск программы Main выглядят следующим образом. (рис.4.1)

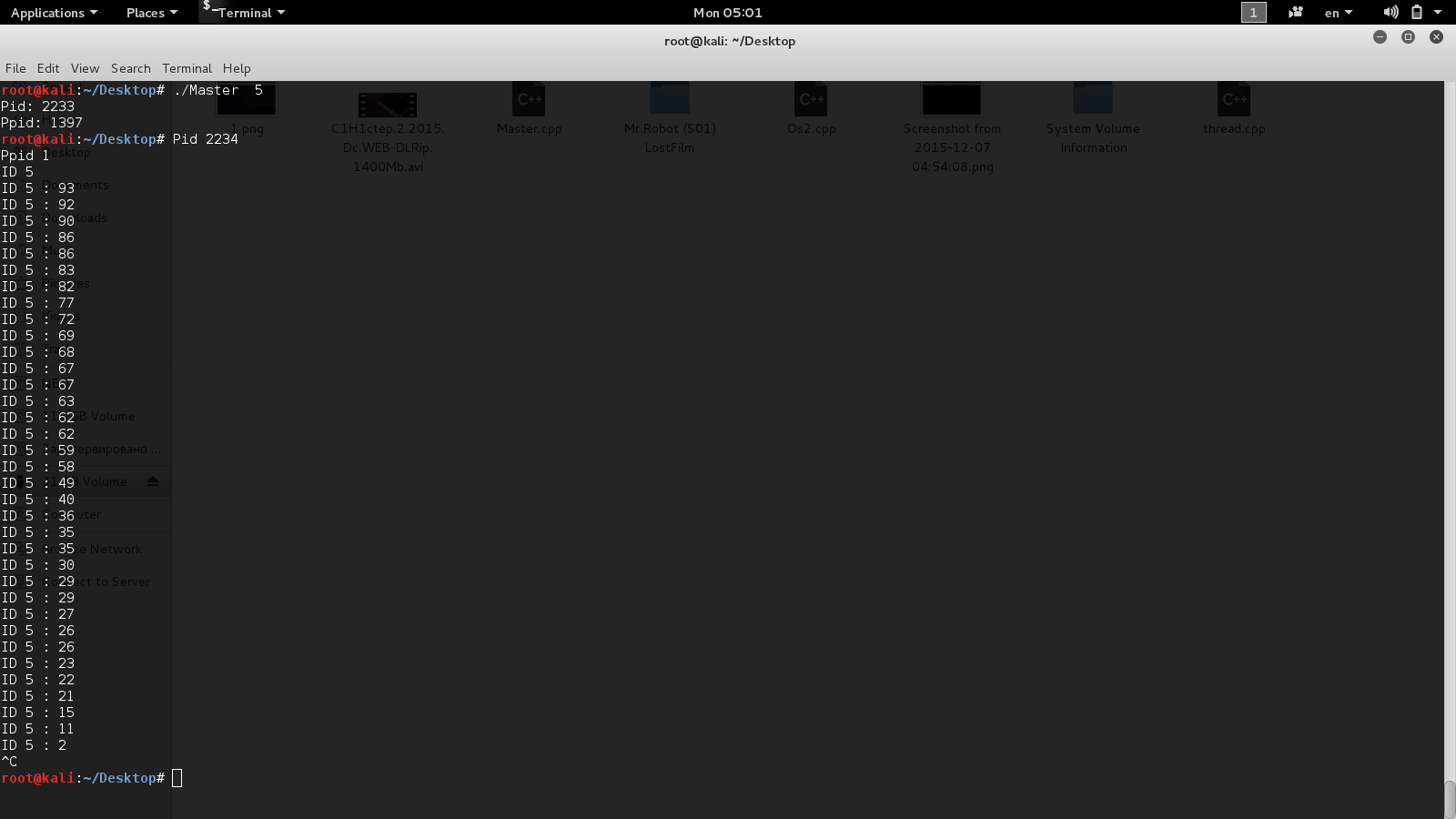


Рисунок 4.1. - Тест программы.

Для компиляции файла с потоками необходимо передавать параметр –lpthread, так как эта библиотека не подключается автоматически.

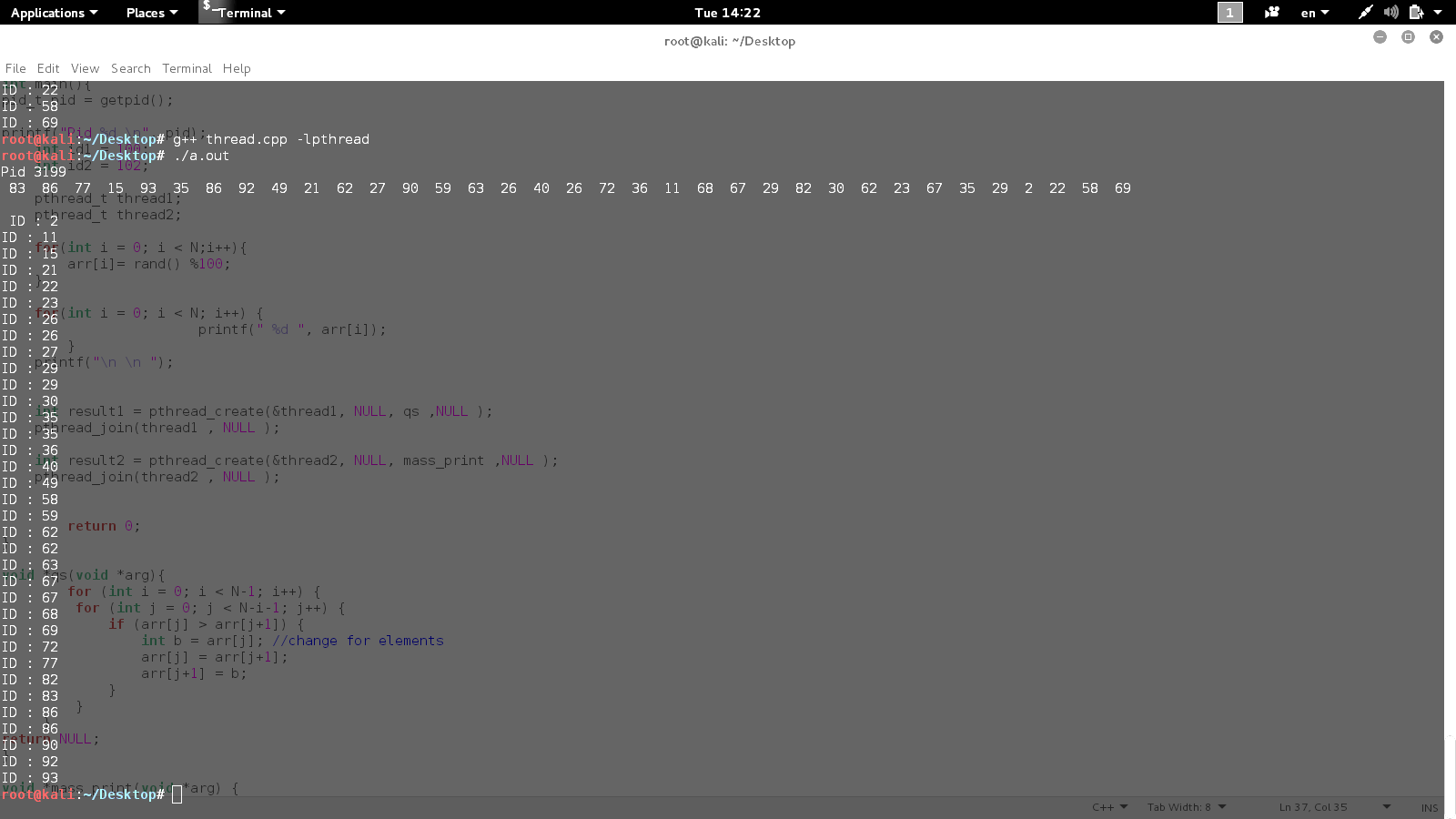


Рисунок 4.2. - Тест программы с потоками.

ВЫВОДЫ

Была написана программа под ОС Linux Ubuntu. Программа работает с потоками и процессами. Было исследовано понятия ID для процессов, потоков. Проанализированы полученные данные