Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

Кафедра ИС

ОТЧЕТ

по лабораторной работе №7

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

Выполнил: ст. гр. И-21д

Куркчи А.Э.

Проверил:

Шишкевич В. Е.

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

Вариант №6

| № варианта | Метод сортировки | Направление  сортировки | N | Функция для выполнения *exec()* | Количество процессов М |
| --- | --- | --- | --- | --- | --- |
| 12 | Шелла | Возр. | 31 | execve() | 2 |

* 1. Написать на языке C программу **Sort**, реализующую следующие действия:
     1. Получить целое число(номер процесса) из первого аргумента программной строки;
     2. Получить с использованием системных вызовов getppid() и getpid() идентификаторы родительского и текущего процесса;
     3. Вывести на экран номер процесса, идентификаторы родительского и текущего процесса;
     4. Заполнить массив целых чисел случайными значениями из диапазона 0-100;
     5. Отсортировать массив;
     6. Вывести на экран отсортированный массив (перед каждым элементом массива выводить номер процесса, полученный в п 3.1.1).

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

* 1. Написать на языке C программу Master, выполняющую следующие действия:
     1. Получить с использованием системных вызовов getppid() и getpid() идентификаторы родительского и текущего процесса и вывести их на экран;
     2. 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)
  2. 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:
     1. Заполнить массив случайными числами;
     2. Используя системные вызовы *pthread\_create()***,** создать потоки **sort** и **mass\_print** и получить идентификаторы созданных потоков.
     3. Вывести на экран идентификаторы созданных потоков (в формате **threads:sort:pthreadId=***threadId***; mass\_print:pthreadId=***threadId*).
     4. Ожидать завершения потоков (с использованием вызова *pthread\_join*).
  3. 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**).

3. ТЕКСТ ПРОГРАММЫ

3.1. Текст программы sort.

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

const int ELS\_NUM **=** 31**;** //количество элементов массива

int gen\_random**(** int range\_min**,** int range\_max **)** **{**

**if** **(**range\_min **>** range\_max**)** **{** //

fprintf**(**stderr**,** "gen\_random(): Invalid arguments\n" **);** exit**(**1**);**

**}**

**return** range\_min**+**rand**()%(**range\_max**-**range\_min**+**1**);**

**}**

void full\_array**(**int **\***arr**,** int size**)** **{** // заполняем массив

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

arr**[**i**]** **=** gen\_random**(**0**,** 100**);**

**}**

**}**

void select\_sort**(**int **\***arr**,** int size**)** **{** // сортировка

int tmp**,** i**,** j**,** pos**;**

**for(**i **=** 0**;** i **<** size**;** **++**i**)** // i - номер текущего шага

**{**

pos **=** i**;**

tmp **=** arr**[**i**];**

**for(**j **=** i **+** 1**;** j **<** size**;** **++**j**)**

**{**

**if** **(**arr**[**j**]** **<** tmp**)**

**{**

pos **=** j**;**

tmp **=** arr**[**j**];**

**}**

**}**

arr**[**pos**]** **=** arr**[**i**];**

arr**[**i**]** **=** tmp**;**

**}**

**}**

void print\_arr**(**int **\***arr**,** int size**)** **{** // вывод массива на экран

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

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

**}**

**}**

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

pid\_t pid**,** ppid**;**

int arr**[**ELS\_NUM**];**

srand**(** time**(** **NULL** **)** **);**

full\_array**(**arr**,** ELS\_NUM**);**

select\_sort**(**arr**,** ELS\_NUM**);**

print\_arr**(**arr**,** ELS\_NUM**);**

**return** 0**;**

**}**

3.2. Текст программы master.

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

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

pid\_t pid**,** ppid**;**

pid **=** getpid **();**

ppid **=** getppid **();**

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

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

**if** **(!**fork **())** **{**

execve **(**"./thread"**,** **NULL,** **NULL);**

fprintf **(**stderr**,** "an error occured\n"**);**

**return** 1**;**

**}**

**return** 0**;**

**}**

3.3. Текст программы thread.

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#define ELS\_NUM 31 //количество элементов массива

int a**[**ELS\_NUM**];**

int gen\_random**(** int range\_min**,** int range\_max **)** **{**

**if** **(**range\_min **>** range\_max**)** **{** //

fprintf**(**stderr**,** "gen\_random(): Invalid arguments\n" **);** exit**(**1**);**

**}**

**return** range\_min**+**rand**()%(**range\_max**-**range\_min**+**1**);**

**}**

void full\_array**(**int **\***arr**,** int size**)** **{** // заполняем массив

int i**;**

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

arr**[**i**]** **=** gen\_random**(**0**,** 100**);**

**}**

**}**

void select\_sort**(**int **\***arr**,** int size**)** **{**

int tmp**,** i**,** j**,** pos**;**

**for(**i **=** 0**;** i **<** size**;** **++**i**)** // i - номер текущего шага

**{**

pos **=** i**;**

tmp **=** arr**[**i**];**

**for(**j **=** i **+** 1**;** j **<** size**;** **++**j**)**

**{**

**if** **(**arr**[**j**]** **<** tmp**)**

**{**

pos **=** j**;**

tmp **=** arr**[**j**];**

**}**

**}**

arr**[**pos**]** **=** arr**[**i**];**

arr**[**i**]** **=** tmp**;**

**}**

**}**

void print\_arr**(**int **\***arr**,** int size**)** **{** // вывод массива на экран

int i**;**

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

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

**}**

printf**(**"\n"**);**

**}**

void **\***sort**()** **{**

pthread\_t sort\_thread\_id**;**

sort\_thread\_id **=** pthread\_self**();**

printf**(**"sort:pthreadId = %d\n"**,** sort\_thread\_id**);**

select\_sort**(**a**,** ELS\_NUM**);**

**return** **NULL;**

**}**

int main**(){**

srand**(** time**(** **NULL** **)** **);**

pthread\_t arr\_print\_thread\_id**;**

pthread\_t thid\_sort**,** thid\_print**,** mythid**;**

int result**;**

result **=** pthread\_create**(** **&**thid\_sort**,**

**(**pthread\_attr\_t **\*)NULL,** sort**,** **NULL);**

**if(**result**){**

printf **(**"Error on thread create, returned value = %d\n"**,** result**);**

exit**(-**1**);**

**}**

full\_array**(**a**,** ELS\_NUM**);**

pthread\_join**(**thid\_sort**,** **(**void **\*\*)NULL);**

arr\_print\_thread\_id **=** pthread\_self**();**

printf**(**"mass\_print:pthreadId = %d\n"**,** arr\_print\_thread\_id**);**

print\_arr**(**a**,** ELS\_NUM**);**

**return** 0**;**

**}**

ВЫВОДЫ

В ходе лабораторной работы изучил понятия процесса и потока ОС семейства UNIX. Приобрел практические навыки разработки программ на языке Си в ОС UNIX , а также написания программ с использованием системных вызовов создания и управления процессами и потоками в ОС UNIX.