Министерство образования и науки Российской Федерации

Новосибирский государственный технический университет

Кафедра программных систем и баз данных

Лабораторная работа №4 по дисциплине

«Управление ресурсами в вычислительных системах»

Факультет: ПМИ

Группа: ПМ-04

Студентки: Анисимова К.Н.

Сулейманова К.А.

Вариант: 4

Преподаватель: Быханов К.В.

Куликов И.М.

Новосибирск

2013

1. ***Цель работы***

Изучить программные средства создания процессов, получить навыки управления и синхронизации процессов, а также простейшие способы обмена данными между процессами. Ознакомиться со средствами динамического запуска программ в рамках порожденного процесса, изучить механизм сигналов ОС UNIX, позволяющий процессам реагировать на различные события, и каналы, как одно из средств обмена информацией между процессами.

1. ***Задание***
2. Разработать программу, реализующую действия, указанные в задании к лабораторной работе с учётом следующих требований:

* все действия, относящиеся как к родительскому процессу, так и к порожденным процессам, выполняются в рамках одного исполняемого файла;
* обмен данными между процессом-отцом и процессом-потомком предлагается выполнить посредством временного файла: процесс-отец после порождения процесса-потомка постоянно опрашивает временный файл, ожидая появления в нем информации от процесса-потомка;
* если процессов-потомков несколько, и все они подготавливают некоторую информацию для процесса-родителя, каждый из процессов помещает в файл некоторую структурированную запись, при этом в этой структурированной записи содержатся сведения о том, какой процесс посылает запись, и сама подготовленная информация.

1. Модифицировать ранее разработанную программу с учётом следующих требований:

* действия процесса-потомка реализуются отдельной программой, запускаемой по одному из системных вызовов execl(), execv() и т.д. из процесса-потомка;
* процесс-потомок, после порождения, должен начинать и завершать свое функционирование по сигналу, посылаемому процессом-предком (это же относится и к нескольким процессам-потомкам);
* обмен данными между процессами необходимо осуществить через программный канал.

Разработать программу, вычисляющую плотность выпуклого распределения в точке х по формуле f(x)=(1-cos(x))/(Pi\*x^2). Для нахождения Pi и cos(х) программа должна породить два параллельных процесса, вычисляющих эти величины путём разложения в ряд по формулам вычислительной математики.

1. ***Решение***

*Алгоритм работы*

* + родительский процесс создает временный файл для передачи данных;
  + родительский процесс порождает два дочерних процесса, запомнив их идентификаторы;
  + родительский процесс постоянно проверяет, не изменился ли временный файл;
  + в это время оба дочерних процесса выполняют свои вычисления и заносят свой идентификатор вместе с результатом в временный файл;
  + обнаружив в файле новые данные, родительский процесс считывает их, и по идентификатору процесса определяет, что за результат он получил;
  + после чего вычисляет значение функции и удаляет временный файл.
  + родительский процесс создает два программных канала;
  + затем порождает два потомка, каждый из которых запускает свою программу (процессы-потомки выполняются параллельно):
    - родительский процесс посылает сигнал потомку и ожидает его завершения;
    - получив сигнал, программа выполняется и записывает результат в канал;
    - после этого родительский процесс читает из канала это значение;
  + родительский процесс читает результаты из каналов и вычисляет значение функции.

Для вычисления Pi использовали ряд Лейбница:

Для вычисления cos(x) использовали ряд Тейлора:

1. ***Средства программирования***

В данной работе были использованы *функции:*

int fork(); - системный вызов fork() используется для порождения нового процесса.

int wait(int \*status); - ожидание завершения процесса-потомка родительским процессом.

int exit(int status); - завершение функционирования процесса

Аргумент status является статусом завершения, который передается отцу процесса, если он выполнял системный вызов wait().

char \*name, \*arg0, \*arg1, ... , \*argn;

execl(name, arg0, arg1, ... , argn, 0) – осуществляется смена программы, определяющей функционирование данного процесса.

1. ***Текст программы***

*Файл main\_3.c*

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

double factor(int n)

{

double s = 1.0;

int i;

for(i=1; i<=n; i++)

s\*=i;

return s;

}

double in\_pow(int n, double x)

{

double s = 1.0;

int i;

for(i=1; i<=n; i++)

s\*=x;

return s;

}

int main()

{

struct storage

{

int pid; //идентификатор потомка

double data; //значение, полученное от потомка

} st;

int i, n, fd, pid\_pi, pid\_cos; //fd - файловый дескриптор .tmp

int a = 1;

double f, x, cos, Pi, s = 1.0;

char \*file\_name = "file.tmp";

struct stat tmp\_stat;

fd = open(file\_name, O\_RDWR); //открываем временый файл

printf("\nВведите x = ");

scanf("%lf", &x);

printf("\nВведите n = ");

scanf("%d", &n);

printf("\n");

pid\_pi = fork();

if (pid\_pi == 0) //первый потомок процесса (для Pi)

{

st.pid = getpid();

st.data = 0.0;

for (i = 0; i < n; i++)

{

st.data += 4.0\*a/(2.0\*i + 1.0);

a = a\*(-1);

}

write (fd, &st, sizeof(st));

exit(0);

}

pid\_cos = fork();

if (pid\_cos == 0) //второй потомок процесса (для cos)

{

st.pid = getpid();

st.data = 1.0;

for (i = 1; i < n; i++)

{

s = factor(2\*i);

a = a\*(-1);

st.data += a\*in\_pow(2\*i,x)/s;

}

write (fd, &st, sizeof(st));

exit(0);

}

do {

fstat(fd, &tmp\_stat);

} while (tmp\_stat.st\_size != 2\*sizeof(st));

lseek(fd,0,SEEK\_SET); //переходим в начало файла

read(fd,&st,sizeof(st));

if (st.pid == pid\_pi)

Pi = st.data;

if (st.pid == pid\_cos)

cos = st.data;

read(fd,&st,sizeof(st));

if (st.pid == pid\_pi)

Pi = st.data;

if (st.pid == pid\_cos)

cos = st.data;

f = (1.0 - cos)/(Pi\*x\*x); //вычисляем функцию

printf("Pi = %lf\n", Pi);

printf("cos(%lf) = %lf\n", x,cos);

printf("f(%lf) = %lf\n", x,f);

close(fd); //закрываем

remove(file\_name); //удаляем

waitpid(pid\_pi,NULL,0);

waitpid(pid\_cos,NULL,0);

sleep(2);

return 0;

}

*Файл new\_main\_3.c*

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <signal.h>

int main()

{

int i, n, pd[2], pid\_pi, pid\_cos, pd2[2]; //переменные для цикла, параметр и для каналов

double f, x, cos, Pi; //параметры и результат

char rpd[10], wpd[10], rpd2[10], wpd2[10]; //чуть-чуть о каналах

//Ввод параметров

printf("\nВведите x = ");

scanf("%lf", &x);

printf("\nВведите n = ");

scanf("%d", &n);

printf("\n");

//cos: создаем канал для узнаём информацию о канале

pipe(pd);

sprintf(rpd,"%d",pd[0]);

sprintf(wpd,"%d",pd[1]);

//cos: передача параметров

write(pd[1], &n, sizeof(n));

write(pd[1], &x, sizeof(x));

//cos: запуск потока

pid\_cos = fork();

if (pid\_cos == 0) //cos: потомок процесса

{

execl("./cos",rpd,wpd,0); //вызываем cos.c

exit(0);

}

//pi: создаем канал для узнаём информацию о канале

pipe(pd2);

sprintf(rpd2,"%d",pd2[0]);

sprintf(wpd2,"%d",pd2[1]);

//pi: передача параметров

write(pd2[1], &n, sizeof(n));

//pi: запуск потока

pid\_pi = fork();

if (pid\_pi == 0) //pi: потомок процесса

{

execl("./pi",rpd2,wpd2,0);//вызываем pi.c

exit(0);

}

waitpid(pid\_pi,NULL,0); //ждем поток пи

read(pd2[0], &Pi, sizeof(Pi)); //получаем pi

printf("Pi after = %lf\n", Pi); //печатаем pi

waitpid(pid\_cos,NULL,0); //ждем поток косинуса

read(pd[0], &cos, sizeof(cos)); //получаем cos

printf("cos(%lf) after = %lf\n", x,cos); //печатаем cos

f = (1.0 - cos)/(Pi\*x\*x); //вычисляем функцию

printf("f(%lf) after = %lf\n", x,f); //печатаем ответ

return 0;

}

*Файл pi.c*

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <signal.h>

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

{

printf("I'm in pi\n");

double pi = 0;

int a = 1;

int n, i;

int pd[2]; //канал для записи дескриптора

sscanf(argv[0], "%d", &pd[0]);

sscanf(argv[1], "%d", &pd[1]);

read(pd[0], &n, sizeof(n));

printf("n in pi = %d\n", n);

for (i = 0; i < n; i++)

{

pi += 4.0\*a/(2.0\*i + 1.0);

a = a\*(-1);

printf("%d %lf\n", i, pi);

}

write (pd[1], &pi, sizeof(pi)); //запись в канал Pipe

printf("Pi in pi = %lf\n", pi);

return 0;

}

*Файл cos.c*

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <signal.h>

//вычисление факториала

double factor(int n)

{

double s = 1.0;

int i;

for(i=1; i<=n; i++)

s\*=i;

return s;

}

//вычисление числа x^n

double in\_pow(int n, double x)

{

double s = 1.0;

int i;

for(i=1; i<=n; i++)

s\*=x;

return s;

}

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

{

printf("I'm in cos\n");

double cos = 1.0; // Значение косинуса

double x; // аргумент косинуса

double s; // переменная хранящая факториал

int a = 1;

int pd[2]; //канал для записи дескриптора

int n, i;

//получаем информацию о канале

sscanf(argv[0], "%d", &pd[0]);

sscanf(argv[1], "%d", &pd[1]);

// считываем параметры

read(pd[0], &n, sizeof(n));

read(pd[0], &x, sizeof(x));

printf("n in cos = %d\n", n);

printf("x in cos = %lf\n", x);

//собственно вычисления

for (i = 1; i < n; i++)

{

s = factor(2\*i);

a = a\*(-1);

cos += a\*in\_pow(2\*i,x)/s;

printf("%d %lf\n", i, cos);

}

write(pd[1], &cos, sizeof(cos)); // возвращаем результат в канал

printf("cos(%lf) in cos = %lf\n", x,cos);//пишем результат на экран

return 0;

}

1. ***Тесты***

Введите x = 1

Введите n = 20

Pi = 3.091624

cos(1.000000) = 0.540302

f(1.000000) = 0.148691



Введите x = 3.14

Введите n = 15

Pi = 3.208186

cos(3.140000) = -0.999999

f(3.140000) = 0.063228



Введите x = 1

Введите n = 20

Pi after = 3.091624

cos(1.000000) after = 0.540302

f(1.000000) after = 0.148691



Введите x = 3.14

Введите n = 15

Pi after = 3.208186

cos(3.140000) after = -0.999999

f(3.140000) after = 0.063228