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 прикладной математики и информатики

Группа ПМ-01

Студенты Курочкин А.В.

Преподаватели Быханов К.В.

Саутин А.С.

Вариант 7

Новосибирск

2013

Постановка задачи

Разработать программу, вычисляющую число сочетаний C(k,n)=n!/(k!\*(n-k)!). Для вычисления факториалов n!, k!, (n-k)! должны быть порождены три параллельных процесса-потомка.

Описание метода решения задачи

1. 1) Выполнить проверку на корректность данных.

2) Создать/открыть файл для связи по данным.

3) Создавать дочерние процессы, считающие факториалы и записывающие результат в файл, если он достиг нужного объёма.

4) Опрашивать файл, если достиг объёма, при котором данных достаточно – продолжить.

5) Выполнить умножение и деление, выдать результат.

1. 1) Выполнить проверку на корректность данных.

2) Создать каналы для прямой и обратной связи.

3) В канал прямой связи поместить все необходимые данные.

4) Запустить дочерние процессы.

5) Подавать сигналы дочерним процессам, по получении которых они производят запись в канал обратной связи и подают сигнал родительскому процессу на считывание и продолжение подачи сигналов остальным дочерним процессам.

6) Завершить вычисления, выдать результат.

Описание программного средства

Программа написана на языке С++, может быть скомпилирована компилятором языка С++, с помощью make-файлов, либо без них.

Запуск программы осуществляется командой:

./upr340

По умолчанию вывод происходит на экран, но может быть легко перенаправлен в файл стандартными средствами интерпретаторов.

Исходный текст

I.

/\*\*

\* @file main.cpp

\* @detailed The programme that able to calculate C(k,n)=n!/(k!\*(n-k)!)

\* @author Kurochkin A.V.

\* @version 0.1

\*/

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <signal.h>

#include <unistd.h>

#include <fcntl.h>

#include <stdlib.h>

/\*\*

\* A structure that contains an id of a proccess

\* and the result of its execution

\*/

struct record

{

int procdescr;//!<An id

long factorial;//!<The result of factorial calculation

} rec1;

/\*\*

\* @brief The procedure of calculation of a factorial

\* @param A file

\* @param The argument of factorial-function

\*/

void process (int tempfile, long temp)

{

long i, factorial;

factorial = 1;

for(i = 2; i <= temp; i++)//!<Calculation of factorial

factorial \*= i;

//!<Now writing the result in a temporary file

rec1.factorial = factorial;

rec1.procdescr = getpid ();

write(tempfile, &rec1, sizeof rec1);

}

/\*\*

\* The main function

\* @param Number of arguments

\* @param n, k

\* @return 1 - Invalid number of arguments

\* 2 - Invalid type of arguments

\* 3 - Arguments are out of range [1,12]

\* 4 - Invalid arguments: n < k

\* 0 - If success

\*/

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

{

int pdn, pdk, pdn\_k, tempfile;

long n, k, temp, fact\_n, fact\_k, fact\_n\_k;

struct stat statbuf;

//<! Data control

if (argc != 3)

{

printf("Invalid number of arguments, try again!\nUsage: %s n k\n", argv[0]);

exit (1);

}

if (!sscanf(argv[1], "%ld", &n) || !sscanf(argv[2], "%ld", &k))

{

printf("Invalid type of arguments!\n");

exit (2);

}

if (n > 12 || n < 1 || k > 12 || k < 1)

{

printf("Arguments are out of range [1,12]!\n");

exit (3);

}

if (n < k)

{

printf("Invalid arguments: n < k!\n");

exit (4);

}

temp = n;

//<! Open a temporary file

tempfile = open("temp", O\_CREAT|O\_RDWR, 0x0777);

//<! A child-1 creation

pdn = fork();

//<! If a child

if(pdn == 0)

{

process(tempfile, temp);

exit (0);

}

//<! A child-2 creation

temp = k;

pdk = fork();

//<! If a child

if(pdk == 0)

{

do

{

fstat(tempfile, &statbuf);

}

while(statbuf.st\_size != sizeof (struct record));

process(tempfile, temp);

exit (0);

}

//<! A child-3 creation

temp = n - k;

pdn\_k = fork ();

//<! If a child

if(pdn\_k == 0)

{

do

{

fstat(tempfile, &statbuf);

}

while(statbuf.st\_size != sizeof (struct record) \* 2);

process(tempfile, temp);

exit (0);

}

//<! When children had been written all the results in the file

do

{

fstat(tempfile, &statbuf);

}

while(statbuf.st\_size != sizeof (struct record) \* 3);

//<! We`d go again all the file

lseek(tempfile, 0, SEEK\_SET);

//<! Then we read this file

for(int ij = 1; ij <= 3;ij++)

{

while(!read(tempfile, &rec1, sizeof rec1));

if(rec1.procdescr == pdn) fact\_n = rec1.factorial;

if(rec1.procdescr == pdk) fact\_k = rec1.factorial;

if(rec1.procdescr == pdn\_k) fact\_n\_k = rec1.factorial;

}

//<! The result output

close(tempfile);

printf("%ld\n", fact\_n / (fact\_k \* fact\_n\_k));

remove("temp");

sleep(1);

return 0;

}

/\*\*

\* @file main.cpp

\* @detailed The programme that able to calculate C(k,n)=n!/(k!\*(n-k)!)

\* @author Kurochkin A.V.

\* @author Popkov I.V.

\* @version 0.1

\*/

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <signal.h>

#include <unistd.h>

#include <fcntl.h>

#include <stdlib.h>

#include <wait.h>

#define SIGNAL\_REPORT\_SEND SIGUSR1

#define SIGNAL\_ORDER\_SEND SIGUSR2

/\*\*

\* @brief The procedure of signal-ignore

\*/

void receive(int){}

/\*\*

\* The main function

\* @param Number of arguments

\* @param n, k

\* @return 1 - Invalid number of arguments

\* 2 - Invalid type of arguments

\* 3 - Arguments are out of range [1,12]

\* 4 - Invalid arguments: n < k

\* 0 - If success

\*/

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

{

int pdn, pdk, pdn\_k, pdres;

long n, k, n\_k;

int direct[2], ret[2];

char str0[10], str1[10];

signal(SIGNAL\_REPORT\_SEND, receive);

//<! Data control

if (argc != 3)

{

printf("Invalid number of arguments, try again!\nUsage: %s n k\n", argv[0]);

exit (1);

}

if (!sscanf(argv[1], "%ld", &n) || !sscanf(argv[2], "%ld", &k))

{

printf("Invalid type of arguments!\n");

exit (2);

}

if (n > 12 || n < 1 || k > 12 || k < 1)

{

printf("Arguments are out of range [1,12]!\n");

exit (3);

}

if (n < k)

{

printf("Invalid arguments: n < k!\n");

exit (4);

}

//!< Pipes creation

pipe(direct);

pipe(ret);

//!< Prepare numbers of descriptors to send it to children

sprintf(str0,"%d",direct[0]);

sprintf(str1,"%d",ret[1]);

//!< Writing in pipe from parent

write(direct[1],&n,sizeof(long));

write(direct[1],&k,sizeof(long));

n\_k = n - k;

write(direct[1],&n\_k,sizeof(long));

//<! A child-1 creation

pdn = fork();

if(pdn == 0)

{

close(direct[1]);

close(ret[0]);

execl("./process",str0,str1,0);

exit(0);

}

//<! A child-2 creation

pdk = fork();

if(pdk == 0)

{

close(direct[1]);

close(ret[0]);

execl("./process",str0,str1,0);

exit(0);

}

//<! A child-3 creation

pdn\_k = fork();

if(pdn\_k == 0)

{

close(direct[1]);

close(ret[0]);

execl("./process",str0,str1,0);

exit(0);

}

//<! Wait a time for passing to signal();

sleep(1);

//<! Induce childs to execute by signals

kill(pdn, SIGNAL\_ORDER\_SEND);

sigpause(SIGNAL\_REPORT\_SEND);

read(ret[0],&n,sizeof(long));

kill(pdk, SIGNAL\_ORDER\_SEND);

sigpause(SIGNAL\_REPORT\_SEND);

read(ret[0],&k,sizeof(long));

kill(pdn\_k, SIGNAL\_ORDER\_SEND);

sigpause(SIGNAL\_REPORT\_SEND);

read(ret[0],&n\_k,sizeof(long));

printf ("%ld\n", n / (k \* n\_k));

return 0;

}

/\*\*

\* @file process.cpp

\* @detailed The programme that able to calculate factorial n!

\* @author Kurochkin A.V.

\* @version 0.1

\*/

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <signal.h>

#include <unistd.h>

#include <fcntl.h>

#include <stdlib.h>

#include <wait.h>

#define SIGNAL\_REPORT\_SEND SIGUSR1

#define SIGNAL\_ORDER\_SEND SIGUSR2

/\*\*

\* @brief The procedure of signal-ignore

\*/

void send(int){}

/\*\*

\* The main function

\* @param Number of arguments

\* @param n

\*/

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

{

int direct, ret;

long n;

long res;

signal(SIGNAL\_ORDER\_SEND, send);

//<! Waiting before be able to execute

sigpause(SIGNAL\_ORDER\_SEND);

direct = atoi(argv[0]);

ret = atoi(argv[1]);

//<! Read from pipe

read(direct,&n,sizeof(long));

res = 1;

//<! Calculation

for(int i = 2; i <= n; i++)

res = res \* i;

//<! Returns result with a pipe

write(ret,&res,sizeof(long));

//<! Signal that the data in the channel

kill(getppid(),SIGNAL\_REPORT\_SEND);

return 0;

}

Набор тестов

**Тест № 1**

./upr470 1 1

Результат:

1

**Тест № 2**

./upr470 12 12

Результат:

1

**Тест № 3**

./upr470 6 2

Результат:

15

**Тест № 4**

./upr470 8 3

Результат:

56

**Тест № 5**

./upr470

Результат:

Invalid number of arguments, try again!

Usage: ./upr470 n k

**Тест № 6**

./upr470 3

Результат:

Invalid number of arguments, try again!

Usage: ./upr470 n k

**Тест № 7**

./upr470 8 5 2

Результат:

Invalid number of arguments, try again!

Usage: ./upr470 n k

**Тест № 8**

./upr470 uf pf

Результат:

Invalid type of arguments!

**Тест № 9**

./upr470 14 8

Результат:

Arguments are out of range [1,12]!

**Тест № 10**

./upr470 9 10

Результат:

Invalid arguments: n < k!