

# **Sprawozdanie z przedmiotu Programowanie Równoległe i Rozproszone**

## **Mechanizmy programowania współbieżnego w SO UNIX/LINUX**

Wykonał:

Imię i nazwisko

Magdalena Paszko

Nr indeksu

76024

Grupa

L2

Prowadzący : dr inż. Krzysztof Szerszeń

# 1. Introduction

The fork () system call is one of the most important functions in Unix systems, including Linux. For her, and only for her, new processes may appear in the system. The only process that is brought to life in any other way is the process with number (id) 1, which is init.

From a programmer's point of view, fork () is a very simple function. It is called like this:

```
pid = fork();
```

and gets two different values as a result. The parent process that triggered the fork gets the child ID. The child scores a 0. This allows processes to know "who is which" without resorting to tricks. The child is almost an exact copy of the parent - it differs only in what fork () returns. Of course, in case of failure, the child is not created and the parent receives information about the error.

## 2. Definition

### Numerical integration

A method based on approximate calculation of definite integrals.

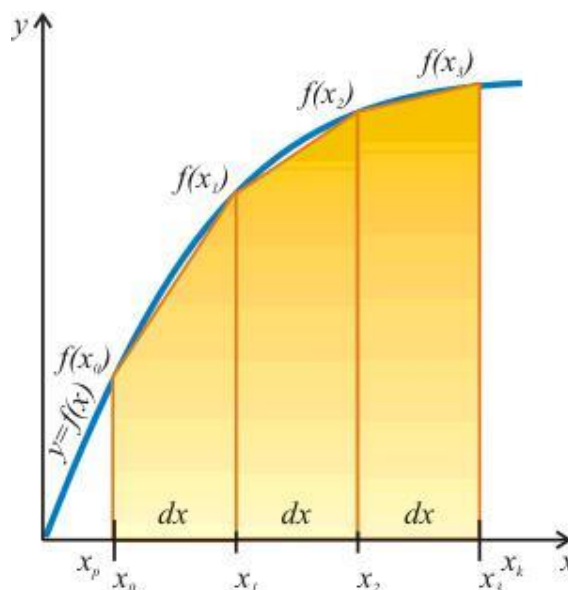
Simple methods of numerical integration rely on the approximation of the integral by means of the appropriate sum of the weighted value of the integral function at several points.

For a more accurate approximation, the integration interval is divided into small pieces.

The final result is the sum of the estimates of the integrals in each sub-interval.

The interval is usually divided into equal sub-intervals.

### The trapezoidal method



The integration interval  $\langle x_p, x_k \rangle$  is divided into  $n + 1$  equally distant points  $x_0, x_1, x_2, \dots, x_n$ . These points we determine in a simple way according to the formula:

for  $i = 0, 1, 2, \dots, n$

$$x_i = x_p + \frac{i}{n}(x_k - x_p)$$

We calculate the distance between two adjacent points - this will be the height of each trapezoid:

$$dx = \frac{x_k - x_p}{n}$$

For each point determined in this way, we calculate the value of the function  $f(x)$  at that point:

$$f_i = f(x_i), \text{ dla } i = 1, 2, \dots, n$$

The area under the function graph is approximated by the  $n$  trapezoidal areas. The area of the  $i$ -th trapezoid is calculated according to the formula:

for  $i=1, 2, \dots, n$

$$f_i = f(x_i), \text{ dla } i = 1, 2, \dots, n$$

The approximate value of the integral is the sum of the fields of all the trapeziums thus obtained:

$$s = P_1 + P_2 + \dots + P_n$$

so:

$$s = \frac{f_0 + f_1}{2} dx + \frac{f_1 + f_2}{2} dx + \frac{f_2 + f_3}{2} dx + \dots + \frac{f_{n-2} + f_{n-1}}{2} dx + \frac{f_{n-1} + f_n}{2} dx$$

$$s = \frac{dx}{2} (f_0 + f_1 + f_1 + f_2 + f_2 + f_3 + \dots + f_{n-2} + f_{n-1} + f_{n-1} + f_n)$$

$$s = \frac{dx}{2} (f_0 + 2f_1 + 2f_2 + \dots + 2f_{n-1} + f_n)$$

$$s = dx (f_1 + f_2 + \dots + f_{n-1} + \frac{f_0 + f_n}{2})$$

The formula at the end is the basis for the approximate calculation of the integral in the trapezoidal method.

$$\int_{x_p}^{x_k} f(x) dx \approx \frac{x_k - x_p}{n} \left( \sum_{i=1}^{n-1} f(x_p + i \frac{x_k - x_p}{n}) + \frac{f(x_p) + f(x_k)}{2} \right)$$

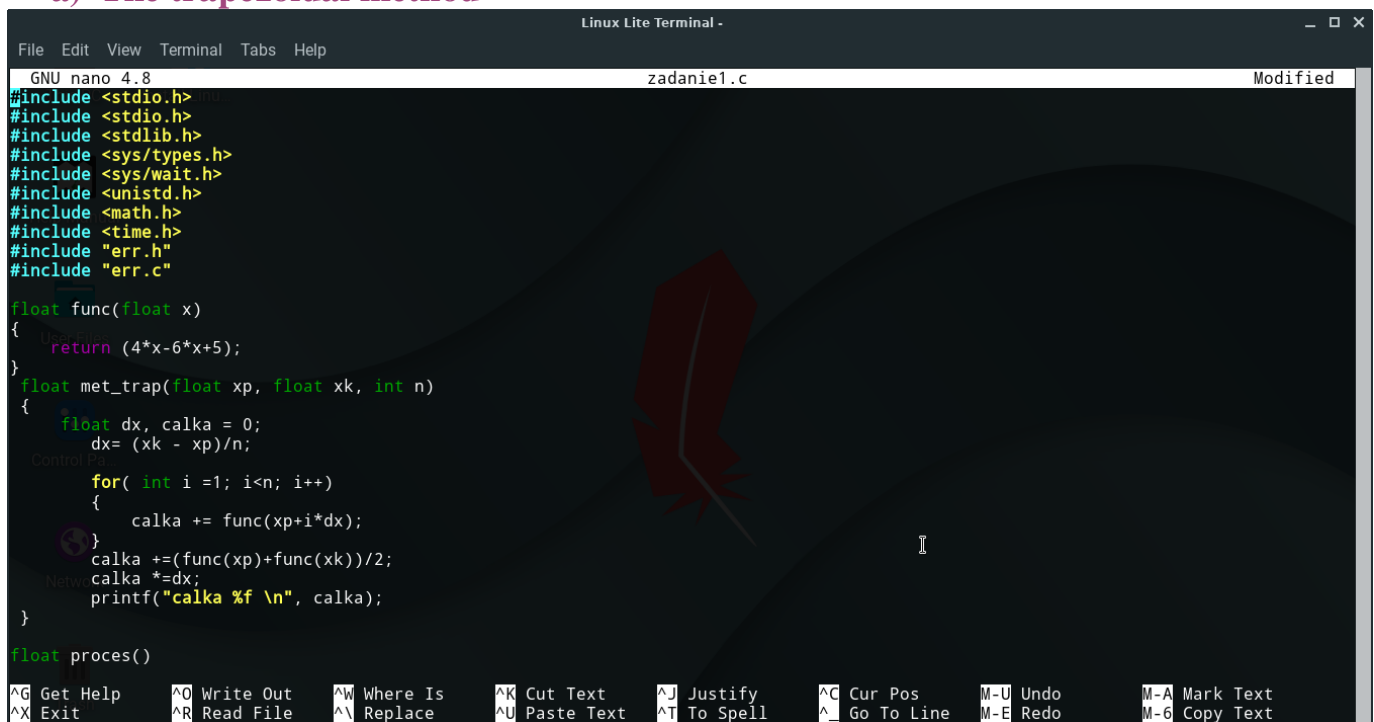
## Model Leibniza

The formula for calculating the nth derivative of the product of a function. It was introduced by the German mathematician Gottfried Leibniz.

$$\pi = 4 \cdot \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{2 \cdot n - 1} = 4 \cdot \left( 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots \right)$$

### 3. Implementation of algorithms

#### a) The trapezoidal method

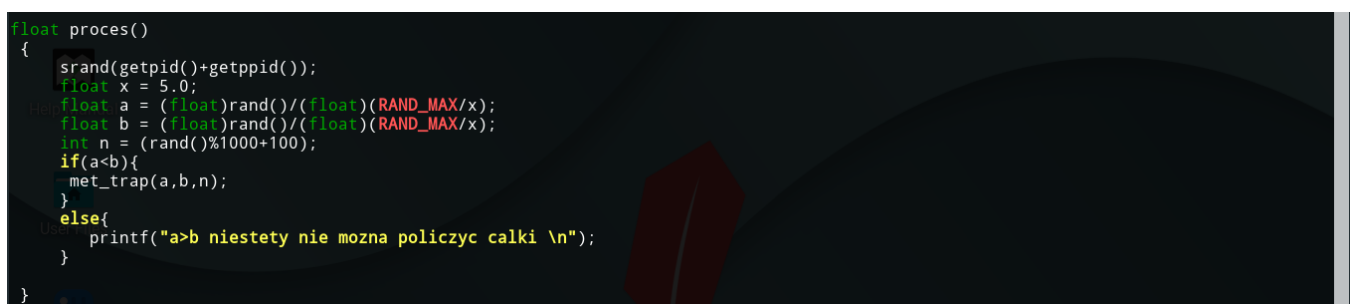


```
GNU nano 4.8 zadanie1.c Modified
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <math.h>
#include <time.h>
#include "err.h"
#include "err.c"

float func(float x)
{
    return (4*x-6*x+5);
}
float met_trap(float xp, float xk, int n)
{
    float dx, calka = 0;
    dx= (xk - xp)/n;
    for( int i =1; i<n; i++)
    {
        calka += func(xp+i*dx);
    }
    calka +=(func(xp)+func(xk))/2;
    calka *=dx;
    printf("calka %f \n", calka);
}

float proces()
{
    srand(getpid()+getppid());
    float x = 5.0;
    float a = (float)rand()/(float)(RAND_MAX/x);
    float b = (float)rand()/(float)(RAND_MAX/x);
    int n = (rand()%1000+100);
    if(a<b){
        met_trap(a,b,n);
    }
    else{
        printf("a>b niestety nie mozna policzyc calki \n");
    }
}
```

Function met\_trap takes a range <a,b> and n the number of trapezoids to compute the integral. Calculates the integral from the trapezoidal method formula I described above.



```
float proces()
{
    srand(getpid()+getppid());
    float x = 5.0;
    float a = (float)rand()/(float)(RAND_MAX/x);
    float b = (float)rand()/(float)(RAND_MAX/x);
    int n = (rand()%1000+100);
    if(a<b){
        met_trap(a,b,n);
    }
    else{
        printf("a>b niestety nie mozna policzyc calki \n");
    }
}
```

The process function sets the starting point which is used to generate a series of pseudo random numbers, however inserting NULL and calling threads the number was always the same. The ideal solution was to have a different seed for each program call. One way is to use srand (getpid ()); where getpid () returns the ID of the current process. However, a better seed can be obtained by combining getpid () which is the parent identifier: srand (getpid () + getppid ()). Then rand returns a pseudo-random floating point number a and b and an integer n, and checks if a < b if it's correct then calls met\_tap.

```
Linux Lite Terminal -
File Edit View Terminal Tabs Help
GNU nano 4.8 zadanie1.c

int main ()
{
printf("Prosze podac ilosc procesow: ");
int ilosc;
scanf("%d", &ilosc);

pid_t pid;
int i;
/* tworzy nowe procesy */
for (i = 1; i <= ilosc; i++)
switch (pid = fork()) {
case -1: /* blad */
syserr("Error in fork\n");
case 0: /* proces potomny */
printf(" Proces %d rozpoczyna prace \n ",i);
proces();
return 0;
}

/* czeka na zakonczenie procesow potomnych */
for (i = 1; i <= ilosc; i++) {
if (wait(0) == -1){
syserr("Error in wait\n");
}
}
return 0;
}

^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text      ^J Justify       ^C Cur Pos      M-U Undo        M-A Mark Text
^X Exit          ^R Read File     ^\ Replace       ^U Paste Text    ^T To Spell      ^_ Go To Line    M-E Redo        M-6 Copy Text
```

In main, as you can see from the comments, we must first specify the number of processes to perform. We create new processes.

If the return value is -1, it means that the process checking the return value is the parent process, but the child process could not be created - error.

If the return value is 0, it means that the process checking the returned value is a child process in which the process functions and they met `_trapezow` - a child process.

Finally, we have wait functions, which means waiting for the child process to end. The function returns the id (pid) of the process which terminated. When it returns -1 it means that some process has not finished.

### Program result:

```
linux ~ > Documents gcc zadanie1.c -o zadanie1 -lm
linux ~ > Documents ./zadanie1
Prosze podac ilosc procesow: 10
Proces 4 rozpoczyna prace
Proces 7 rozpoczyna prace
Proces 3 rozpoczyna prace
calka 0.792679
Proces 5 rozpoczyna prace
Proces 2 rozpoczyna prace
Proces 6 rozpoczyna prace
calka 1.247457
calka -0.471698
Proces 8 rozpoczyna prace
a>b niestety nie mozna policzyc calki
a>b niestety nie mozna policzyc calki
calka -3.239468
Proces 9 rozpoczyna prace
Proces 10 rozpoczyna prace
calka 1.652073
Proces 1 rozpoczyna prace
a>b niestety nie mozna policzyc calki
calka -3.844780
calka 4.108016
linux ~ > Documents
```

## b) Model Leibniza

```
Linux Lite Terminal -
File Edit View Terminal Tabs Help
GNU nano 4.8 program2.c Modified
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include "err.h"
#include "err.c"
#include <math.h>
#include <time.h>
#include <sys/mman.h>

float lib()
{
    srand(getpid()+getppid());
    int n=0;
    n= (rand()%4000+100);
    float wynik = 0, liczba = 0, zmienna = 0;
    for(int i=0; i<=n; i++){
        liczba =(pow((- 1.0), i ) ) / ( 2 * i + 1 );
        zmienna +=liczba;
    }
    wynik = 4*zmienna;
    return wynik;
}

int proces()
{
    printf("Liczba PI wynosi: %f \n", lib());
}

int main ()
{
    // ...
}
```

The lib function sets the starting point which is used to generate a series of pseudo random numbers similar to the first example. Calculates the approximation of the number PI by the Leibniz formula to the number that is generated pseudorandom, returning the result of the approximation at the end.

```
Linux Lite Terminal -
File Edit View Terminal Tabs Help
GNU nano 4.8 program2.c Modified
}

int main ()
{
    printf("Prosze podac ilosc procesow: ");
    int ilosc;
    scanf("%d", &ilosc);

    pid_t pid;
    int i;
    /* tworzy nowe procesy */
    for (i = 1; i <= ilosc; i++)
        switch (pid = fork()) {
            case -1: /* blad */
                syserr("Error in fork\n");

            case 0: /* proces potomny */
                printf(" Proces %d rozpoczyna prace \n ",i);
                proces();
                return 0;
        }

    /* czeka na zakonczenie procesow potomnych */
    for (i = 1; i <= ilosc; i++) {
        if (wait(0) == -1){
            syserr("Error in wait\n");
        }
    }
    return 0;
}
```

Main is the same as in the first example, the process function calls the lib function which calculates the PI number

## Program result:

```
linux ~ > Documents nano program2.c
linux ~ > Documents gcc program2.c -o program2 -lm
linux ~ > Documents ./program2
Proszę podać ilość procesów: 11
Proces 3 rozpoczyna prace
Proces 4 rozpoczyna prace
Liczba PI wynosi: 3.144426
Proces 2 rozpoczyna prace
Proces 5 rozpoczyna prace
Liczba PI wynosi: 3.141004
Liczba PI wynosi: 3.141116
Proces 6 rozpoczyna prace
Liczba PI wynosi: 3.141204
Proces 7 rozpoczyna prace
Liczba PI wynosi: 3.141850
Proces 8 rozpoczyna prace
Proces 9 rozpoczyna prace
Proces 10 rozpoczyna prace
Liczba PI wynosi: 3.140181
Proces 1 rozpoczyna prace
Proces 11 rozpoczyna prace
Liczba PI wynosi: 3.142068
Liczba PI wynosi: 3.141991
Liczba PI wynosi: 3.141978
Liczba PI wynosi: 3.140997
Liczba PI wynosi: 3.141329
linux ~ > Documents
```