LAB 9 main()



* Threads->way more easier to use



* Pass references between the threads



thread



Schema:



->main process->master->executes code



->start with a simple c program which prints “hello world!” Wait()A screen shot of a computer screen

Description automatically generated with low confidence

A screen shot of a computer

Description automatically generated with medium confidence

->task: create a number of threads

p- number of processes

!! always add #include <pthread.h> ->store the reference->for later use:

phread\_t vector[of size p]

->create the threads:pthread\_create(&th[i], NULL, f, NULL);

->third argument is a function for “pthread\_create”

->create a function above the main that uses pthread\_self()

->pthread\_join()

#include<stdio.h>

#include<pthread.h>

A picture containing text, screenshot, software, font

Description automatically generated

->compile:

A picture containing text, screenshot, font

Description automatically generated

->extend the solution:

->ids[i] ->pointer type int !!but it has to be of type void

->(void\*)&ids[i] ->cast it as of void type

-> in the f function we have to derefence it as an int: int id=\*(int\*)arg;

A picture containing text, screenshot, software

Description automatically generated

A picture containing text, screenshot, font

Description automatically generated

->compile: A screenshot of a computer

Description automatically generated with medium confidence

->implement a problem that calculates the sum of 2 vectors:

A-n numbers,B-n numbers => the sum is vector C

->function executed n times: C[i]=A[i]+B[i]

->divide vectors in n/p intervals ->p-1 maximum remainder

->create a new program from scratch:

A screen shot of a computer program

Description automatically generated with low confidence

->this will just create some threads that do nothing 😊

->declare 3 vectors(A,B,C) of type int and an n(inside the main)

A screen shot of a computer code

Description automatically generated with low confidence

A picture containing text, screenshot, font

Description automatically generated

->add a function that prints a vector on the screen

A computer code on a black background

Description automatically generated with low confidence

->declare 2 var: start=0 and end

->also one for a remainder(r)=n%p

->go to the for iteration

A picture containing text, screenshot, font, software

Description automatically generated

->declare a structure to pass the arguments(outside the main)

A black background with white text

Description automatically generated with low confidence

->add an array of parameters: param params[p];->store the elements passed

A picture containing text, screenshot, font

Description automatically generated

->back to the f function:

A picture containing text, screenshot, font

Description automatically generated

->compile

A screen shot of a computer program

Description automatically generated with low confidence

Whole code:

#include <stdio.h>  
#include <stdlib.h>  
#include <pthread.h>

typedef struct {  
        int \*a, \*b, \*c, start, end;  
} param;

void\* f(void\* arg)  
{  
        param params = \*(param\*)arg;

        for (int i = params.start; i < params.end; i++)  
                params.c[i] = params.a[i] + params.b[i];

        return NULL;  
}

void printArray(int \*arr, int size)  
{  
        for (int i = 0; i < size; i++)  
                printf("%d ", arr[i]);

        printf("\n");  
}

int main()  
{  
        int p = 4;  
        pthread\_t th[p];

        int \*a, \*b, \*c, n = 10;

        a = (int\*) malloc(sizeof(int) \* n);  
        b = (int\*) malloc(sizeof(int) \* n);  
        c = (int\*) malloc(sizeof(int) \* n);

        for (int i = 0; i < n; i++)  
        {  
                a[i] = rand() % 10;  
                b[i] = rand() % 10;  
        }

        printf("a = ");  
        printArray(a, n);

        printf("b = ");  
        printArray(b, n);

        int start = 0, end;  
        int r = n % p;

        param params[p];

        for (int i = 0; i < p; i++)  
        {  
                end = start + n / p;  
                if (r > 0)  
                {  
                        end++;  
                        r--;  
                }

                params[i].a = a; params[i].b = b; params[i].c = c;  
                params[i].start = start; params[i].end = end;

                pthread\_create(&th[i], NULL, f, (void\*)&params[i]);  
                start = end;  
        }

        for (int i = 0; i < p; i++)  
                pthread\_join(th[i], NULL);

        printf("c= ");  
        printArray(c, n);

        return 0;  
}

Method 2:

#include <stdio.h>  
#include <stdlib.h>  
#include <pthread.h>

typedef struct {  
        int \*a, \*b, \*c, id, n, p;  
} param;

void\* f(void\* arg)  
{  
        param params = \*(param\*)arg;

        for (int i = params.id; i < params.n; i = i + params.p)  
                params.c[i] = params.a[i] + params.b[i];

        return NULL;  
}

void printArray(int \*arr, int size)  
{  
        for (int i = 0; i < size; i++)  
                printf("%d ", arr[i]);

        printf("\n");  
}

int main()  
{  
        int p = 4;  
        pthread\_t th[p];

        int \*a, \*b, \*c, n = 10;

        a = (int\*) malloc(sizeof(int) \* n);  
        b = (int\*) malloc(sizeof(int) \* n);  
        c = (int\*) malloc(sizeof(int) \* n);

        for (int i = 0; i < n; i++)  
        {  
                a[i] = rand() % 10;  
                b[i] = rand() % 10;  
        }

        printf("a = ");  
        printArray(a, n);

        printf("b = ");  
        printArray(b, n);

        param params[p];

        for (int i = 0; i < p; i++)  
        {  
                params[i].a = a; params[i].b = b; params[i].c = c;  
                params[i].id = i; params[i].n = n; params[i].p = p;

                pthread\_create(&th[i], NULL, f, (void\*)&params[i]);  
        }

        for (int i = 0; i < p; i++)  
                pthread\_join(th[i], NULL);

        printf("c= ");  
        printArray(c, n);

        return 0;  
}