ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΦΡΟΝΤΙΣΤΗΡΙΟ – ΑΣΚΗΣΗ 1

ΜΠΑΣΑΓΙΑΝΝΗ ΓΕΩΡΓΙΑ
1084016

Για την εκπόνηση της άσκησης τα αρχεία με τους κώδικες στη c γράφτηκαν με χρήση VSCode και για την μεταγλώττιση και εκτέλεση τους χρησιμοποιήθηκε το WSL (μέσω terminal).

Λύση με διεργασίες

C: > Users > georg > Desktop > mathimata > 7o > OS > C f1_processes.c

```c
#include <stdio.h>
#include <stdlib.h> // for atoi, exit
#include <unistd.h> // for fork, sleep
#include <sys/wait.h> // for wait
//--------------------------------

int main(int argc, char* argv[]) {
    int n;
    long int sum=0, N, k, i, j, val;
    int fd[2];

    // create pipe
    pipe(fd);

    // get number of child processes from command line
    if (argc > 1) {
        n = atoi(argv[1]);
        N = atoi(argv[2]);
    }

    for (i=0; i < n; i++) {
        int pid = fork();

        if (pid == 0) { // child process
            sleep(i + 2); //
            // calc a sum
            sum = 0;
            for (k = ((N/n)*i)+1; k <= (N/n)*(i+1); k++)
                sum += k;

            printf("Computed sum by child %ld process is %ld\n", i, sum);

            close(fd[0]); // write, close read
            write(fd[1], &sum, sizeof(sum));

            close(fd[1]); // close write

            exit(0);
        }
        read(fd[0], &val, sizeof(val)); // waiting
        printf("Parent receives value %ld from child %ld\n", val, i);
        sum += val;
    }
        close(fd[0]); // close read

    // parent process, check total sum
    for(j=N; j>N-(N%n); j--)
        sum += j;

    if(sum==N*(N+1)/2)
        printf("The total sum is %ld\n", sum);

    return 0;
}
```

Με χρήση της systemcall atoi ο κώδικας παίρνει τον αριθμό των διεργασιών n, καθώς επίσης και τον ακέραιο N για το άθροισμα από την γραμμή εντολών του terminal. Ορίζεται επίσης δύο περιγραφείς αρχείου fd για να χρησιμοποιηθούν κατά την σωλήνωση pipe(fd) των διεργασιών.

Έτσι, με την systemcall fork δημιουργούμε n διεργασίες παιδιά (for loop) και η κάθε διεργασία παιδί εκτελεί τον αλγόριθμο που φαίνεται στην επόμενη for loop για να υπολογίσει η κάθε μια N/n όρους. Ο συγκεκριμένος αλγόριθμος είναι αρκετά απλός καθώς ξεκινά από N/n(*i) όρους και γίνεται για επιπλέον N/n όρους (*(i+1)). Η πρόσθεση του 1 στην αρχή είναι επειδή το i αρχικοποιείται στο μηδέν ενώ οι διεργασίες n στο ένα.

Όταν υπολογιστεί το προκείμενο άθροισμα με την close(fd[0]) κλείνει το read ώστε να μπορέσει να γράψει με την write το άθροισμα που υπολογίστηκε και μετά κλείνει και το write με την close(fd[1]).

Η διεργασία γονέας διαβάζει με την read το άθροισμα κάθε διεργασίας παιδί και τα προσθέτει όλα μαζί (sum+=val) κι έπειτα κλείνει το read με close(fd[0]).

Το τελευταίο κομμάτι του αθροίσματος φροντίζει να προσθέσει η διεργασία γονέας με τον αλγόριθμο της τελευταίας for loop κατά τον οποίο προσθέτει το N και προς τα κάτω τόσους όρους όσοι είναι και το υπόλοιπο της διαίρεσης N/n (προφανώς όταν δεν υπάρχει υπόλοιπο δεν θα εκτελεστεί καμία φορά αυτή η for).

Τέλος ελέγχει αν επιβεβαιώνεται η σχέση $sum = 1 + 2 + 3 + ... + N = N(N + 1)/2$ με το if ελέγχου στο τέλος, κι αν είναι ορθή η επιβεβαίωση τυπώνεται το τελικό άθροισμα.

Παραδείγματα Εκτέλεσης

```
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ gcc f1_processes.c -o f1_processess
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ ./f1_processes 3 1000000
Computed sum by child 0 process is 55555611111
Parent receives value 55555611111 from child 0
Computed sum by child 1 process is 166666500000
Parent receives value 166666500000 from child 1
Computed sum by child 2 process is 277777388889
Parent receives value 277777388889 from child 2
The total sum is 500000500000
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ gcc f1_processes.c -o f1_processess
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ ./f1_processes 7 20
Computed sum by child 0 process is 3
Parent receives value 3 from child 0
Computed sum by child 1 process is 7
Parent receives value 7 from child 1
Computed sum by child 2 process is 11
Parent receives value 11 from child 2
Computed sum by child 3 process is 15
Parent receives value 15 from child 3
Computed sum by child 4 process is 19
Parent receives value 19 from child 4
Computed sum by child 5 process is 23
Parent receives value 23 from child 5
Computed sum by child 6 process is 27
Parent receives value 27 from child 6
The total sum is 210
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ gcc f1_processes.c -o f1_processess
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ ./f1_processes 3 30
Computed sum by child 0 process is 55
Parent receives value 55 from child 0
Computed sum by child 1 process is 155
Parent receives value 155 from child 1
Computed sum by child 2 process is 255
Parent receives value 255 from child 2
The total sum is 465
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ gcc f1_processes.c -o f1_processess
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ ./f1_processes 4 1500
Computed sum by child 0 process is 70500
Parent receives value 70500 from child 0
Computed sum by child 1 process is 211125
Parent receives value 211125 from child 1
Computed sum by child 2 process is 351750
Parent receives value 351750 from child 2
Computed sum by child 3 process is 492375
Parent receives value 492375 from child 3
The total sum is 1125750
```

Λύση με Νήματα

C: > Users > georg > Desktop > mathimata > 7ο > OS > C f1_threads.c

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

// global variable: accessible to all threads
long int thread_count;
long int N;

// thread function
void *work(void* rank) {
    long tid = (long) rank;
    long int i, v=0;

    for(i=(tid*(N/thread_count))+1; i<=(tid+1)*(N/thread_count); i++)
        v += i;

    printf("thread %ld produces %ld\n", tid, v);
    pthread_exit((void*)v);

}

//--------------------------------
int main(int argc, char* argv[]) {
    long t;
    pthread_t* thread_handles;
    long int j, sum=0;

    // get number of threads from command line
    if (argc > 1){
        thread_count = atoi(argv[1]);
        N = atoi(argv[2]);
    }

    thread_handles = malloc (thread_count * sizeof(pthread_t));

    // create thread_count independent threads
    for (t = 0; t < thread_count; t++)
        pthread_create(&thread_handles[t], NULL, work, (void*) t);

    sleep(1);
    printf("---Hello from the main thread---\n");

    // wait until threads are completed before main continues
    void* r;
    for (t = 0; t < thread_count; t++) {
        pthread_join(thread_handles[t], &r);

        // 1.
        printf("\tthread %ld returns %ld to the main thread\n", t, (long)r);
        sum +=(long)r;
    }

    // main thread, check total sum
    for(j=N; j>N-(N%thread_count); j--)
        sum += j;

    if(sum==N*(N+1)/2)
        printf("The total sum is %ld\n", sum);

    free(thread_handles);
    return 0;
}
```

Αρχικά ορίστηκαν ως global μεταβλητές η thread_count και η N (αντίστοιχες των n και N) για να έχει πρόσβαση σε αυτές και η main(κύριο νήμα) και τα υπόλοιπα threads όταν κάνουν την work τους.

Στην work που είναι η δουλειά που κάνουν τα νήματα είναι ο αλγόριθμος για να υπολογίζει τους N/thread_count αριθμητικούς όρους και κάθε thread επιστρέφει μέσω της v μία τιμή στο κύριο νήμα που είναι το μερικό άθροισμα που έχει υπολογίσει.

Στην main, πάλι με χρήση της atoi λαμβάνονται οι τιμές των παραμέτρων από την γραμμή εντολών. Με την malloc δεσμεύεται όσος χώρος χρειάζονται τα threads στο heap και έπειτα δημιουργούνται με την κλήση της pthread_create.

Η sleep(1) κάνει το σύστημα να περιμένει πρώτα να ολοκληρωθούν τα αθροίσματα των νημάτω v και μετά να τα επιστρέψουν στο κύριο νήμα (αυτό βοηθάει μόνο στην μη σύγχηση των αποτελεσμάτων printf κατά την εκτέλεση).

Αφού γίνουν join όλα τα νήματα για να μπορέσει να επιστραφεί η τιμή v στην οποία δείχνει ο δείκτης r, τότε προστίθενται η v κάθε νήματος στο sum. Έπειτα το κύριο νήμα προσθέτει το τελευταίο μέρος του αθροίσματος και κάνει τον έλεγχο στο αποτέλεσμα,όπως ακριβώς στην προηγούμενη λύση. Τέλος ελευθερώνεται η μνήμη στο heap που είχε δεσμευτεί κατά την δημιουργία των νημάτων.

Παραδείγματα Εκτέλεσης

```
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ ./f1_threads 4 1000000
thread 0 produces 31250125000
thread 2 produces 156250125000
thread 1 produces 93750125000
thread 3 produces 218750125000
---Hello from the main thread---
        thread 0 returns 31250125000 to the main thread
        thread 1 returns 93750125000 to the main thread
        thread 2 returns 156250125000 to the main thread
        thread 3 returns 218750125000 to the main thread
The total sum is 500000500000
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ gcc f1_threads.c -o f1_threads
f1_threads.c: In function ‘main’:
f1_threads.c:40:5: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
   40 |     sleep(1);
      |     ^~~~~
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ ./f1_threads 5 15
thread 0 produces 6
thread 1 produces 15
thread 3 produces 33
thread 2 produces 24
thread 4 produces 42
---Hello from the main thread---
        thread 0 returns 6 to the main thread
        thread 1 returns 15 to the main thread
        thread 2 returns 24 to the main thread
        thread 3 returns 33 to the main thread
        thread 4 returns 42 to the main thread
The total sum is 120
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ gcc f1_threads.c -o f1_threads
f1_threads.c: In function ‘main’:
f1_threads.c:40:5: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
   40 |     sleep(1);
      |     ^~~~~
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ ./f1_threads 2 131
thread 0 produces 2145
thread 1 produces 6370
---Hello from the main thread---
        thread 0 returns 2145 to the main thread
        thread 1 returns 6370 to the main thread
The total sum is 8646
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ gcc f1_threads.c -o f1_threads
f1_threads.c: In function ‘main’:
f1_threads.c:40:5: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
   40 |     sleep(1);
      |     ^~~~~
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$ ./f1_threads 6 3007
thread 0 produces 125751
thread 1 produces 376752
thread 2 produces 627753
thread 3 produces 878754
thread 4 produces 1129755
thread 5 produces 1380756
---Hello from the main thread---
        thread 0 returns 125751 to the main thread
        thread 1 returns 376752 to the main thread
        thread 2 returns 627753 to the main thread
        thread 3 returns 878754 to the main thread
        thread 4 returns 1129755 to the main thread
        thread 5 returns 1380756 to the main thread
The total sum is 4522528
basagianni@LAPTOP-GECQDV8A:/mnt/c/Users/georg/Desktop/Mathimata/7o/OS$
```