Sharif-OS-Lab / summer1403-7-99101087_99100422

Code Issues 1 Pull requests Actions Projects Security Insights

Edit New issue Jump to bottom

Session 7 Report #1

Open 28 tasks done Amirreza81 opened this issue 2 weeks ago · 0 comments

Assignees

Labels documentation

Amirreza81 commented 2 weeks ago • edited

Team Name: `99101087-99100422`

Student Name of member 1: `AmirReza Azari`
Student No. of member 1: `99101087`

Student Name of member 2: `Bozorgmehr Zia`
Student No. of member 2: `99100422`

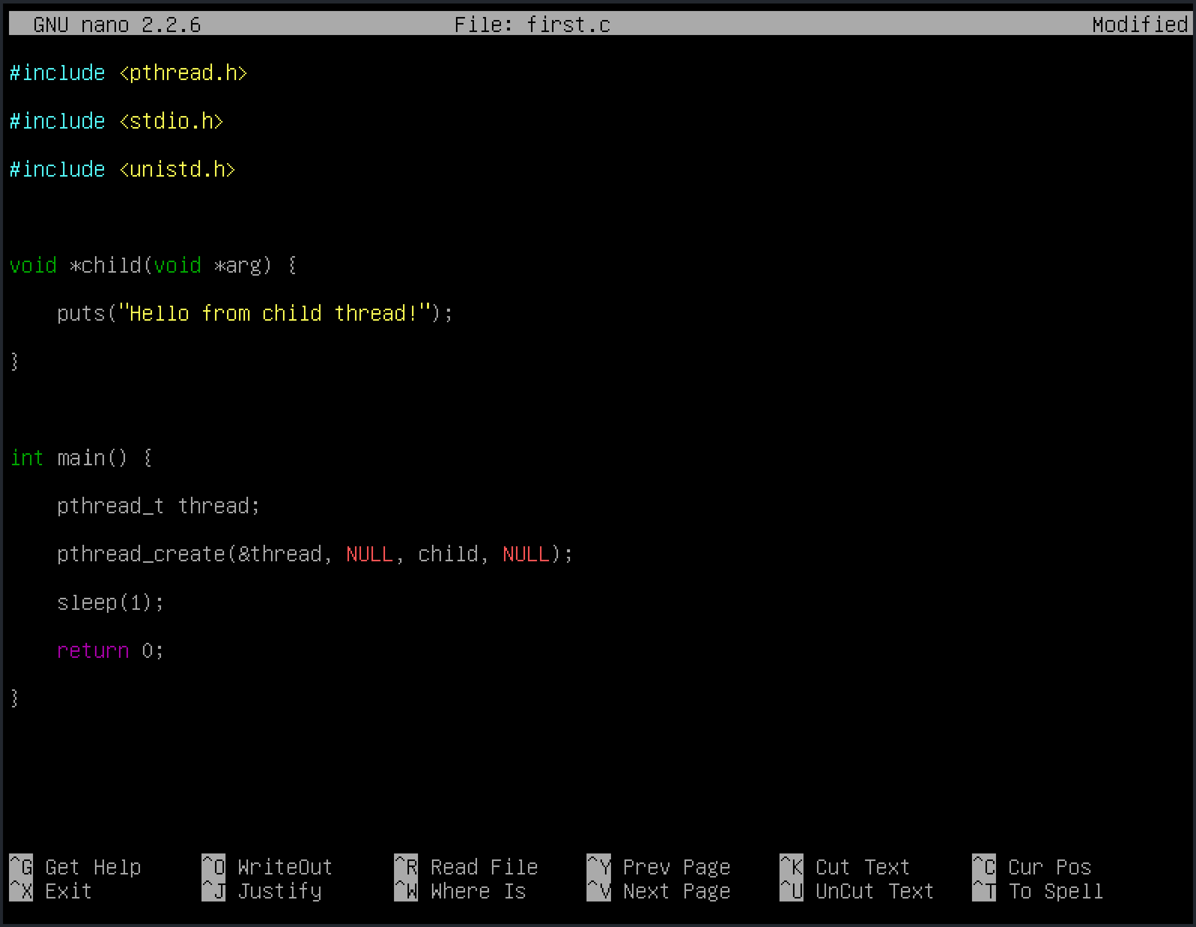
- [x] Read Session Contents.

Section 7.4

Section 7.4.1

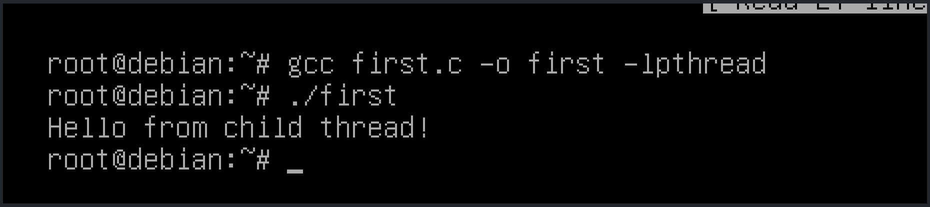
- [x] Creating a thread using pthread

- [x] The code is:



```
  GNU nano 2.2.6                 File: first.c                              Modified

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

void *child(void *arg) {
    puts("Hello from child thread!");
}

int main() {
    pthread_t thread;
    pthread_create(&thread, NULL, child, NULL);
    sleep(1);
    return 0;
}

^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

The result is:

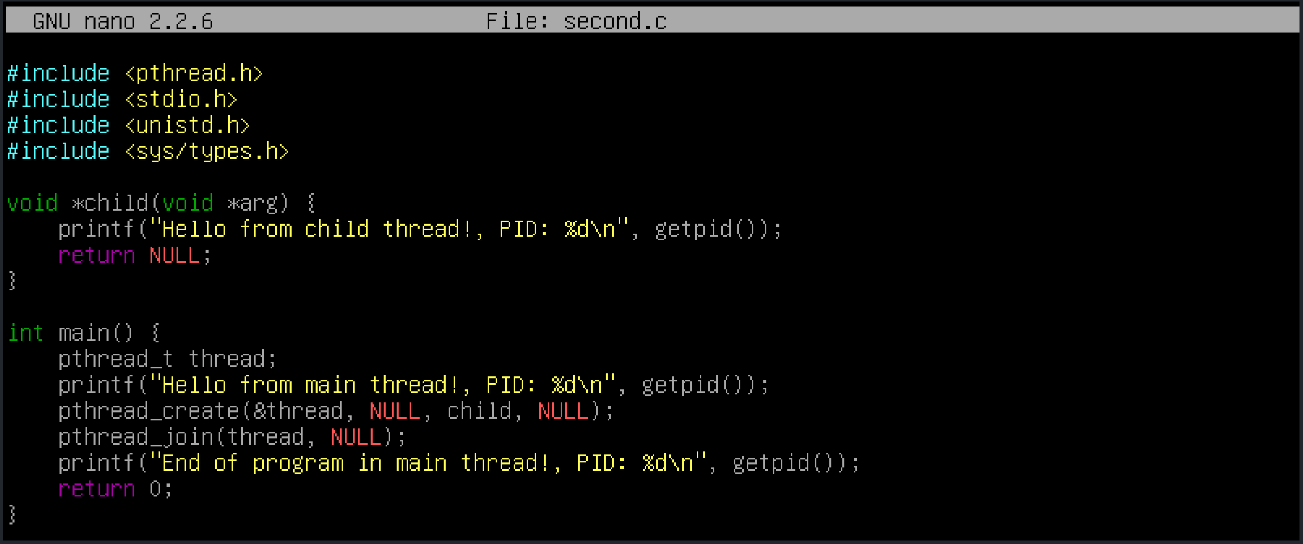


```
root@debian:~# gcc first.c -o first -lpthread
root@debian:~# ./first
Hello from child thread!
root@debian:~# _
```

- [x] Checking the process ids

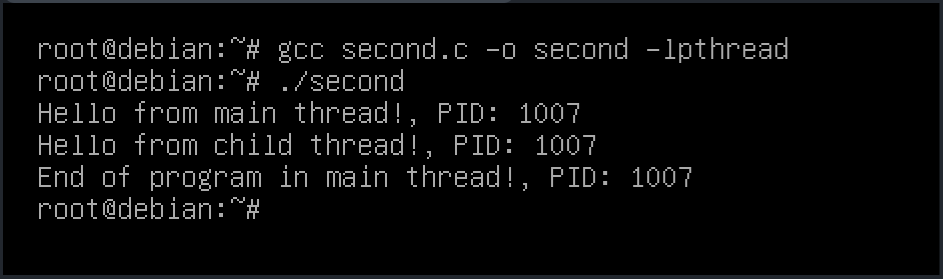
  - [x] `FILL HERE with screenshot of code`

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

void *child(void *arg) {
    printf("Hello from child thread!, PID: %d\n", getpid());
    return NULL;
}

int main() {
    pthread_t thread;
    printf("Hello from main thread!, PID: %d\n", getpid());
    pthread_create(&thread, NULL, child, NULL);
    pthread_join(thread, NULL);
    printf("End of program in main thread!, PID: %d\n", getpid());
    return 0;
}
```



```
GNU nano 2.2.6                    File: second.c

#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

void *child(void *arg) {
    printf("Hello from child thread!, PID: %d\n", getpid());
    return NULL;
}

int main() {
    pthread_t thread;
    printf("Hello from main thread!, PID: %d\n", getpid());
    pthread_create(&thread, NULL, child, NULL);
    pthread_join(thread, NULL);
    printf("End of program in main thread!, PID: %d\n", getpid());
    return 0;
}
```

- [x] `FILL HERE with execution of code`



```
root@debian:~# gcc second.c -o second -lpthread
root@debian:~# ./second
Hello from main thread!, PID: 1007
Hello from child thread!, PID: 1007
End of program in main thread!, PID: 1007
root@debian:~#
```

- [x] `FILL HERE with your descriptions (write in English or Persian)`

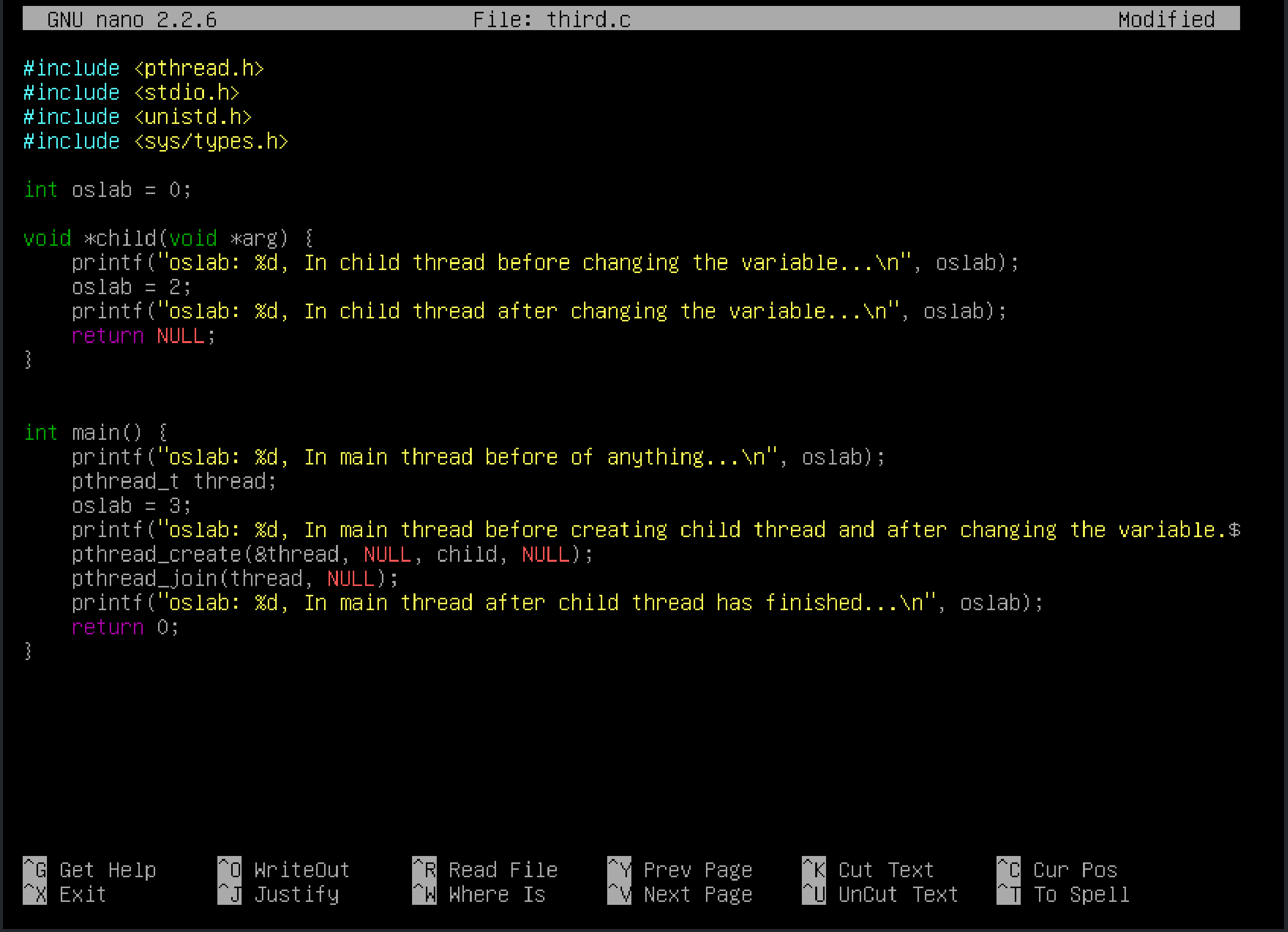
بله، شماره پردازه های چاپ شده در تمامی ریسه ها یکسان خواهد بود. دلیل آن این است که ریسه ها در یک پردازه مشترک اجرا شده و از همان حافظه بهره می‌برند. در نتیجه
PID
آن‌ها یکسان خواهد بود که در اجرای کد نیز این مورد را مشاهده نمودیم.

- [x] Shared variables

- [x] `FILL HERE with screenshot of codes`

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
```

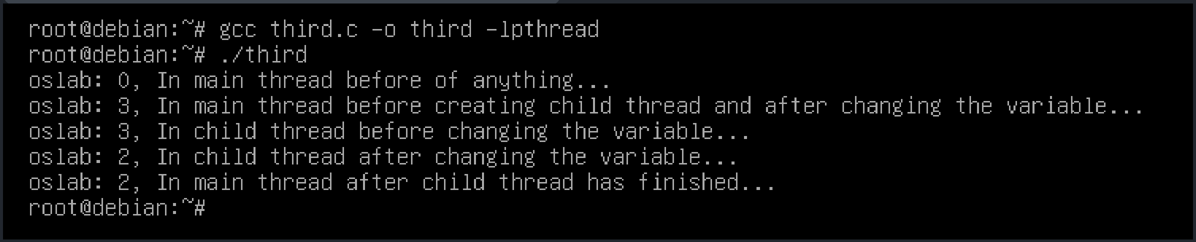
```
#include <sys/types.h>

int oslab = 0;

void *child(void *arg) {
    printf("oslab: %d, In child thread before changing the variable...\n",
oslab);
    oslab = 2;
    printf("oslab: %d, In child thread after changing the variable...\n",
oslab);
    return NULL;
}

int main() {
    printf("oslab: %d, In main thread before of anything...\n", oslab);
    pthread_t thread;
    oslab = 3;
    printf("oslab: %d, In main thread before creating child thread and after
changing the variable...\n", oslab);
    pthread_create(&thread, NULL, child, NULL);
    pthread_join(thread, NULL);
    printf("oslab: %d, In main thread after child thread has finished...\n",
oslab);
    return 0;
}
```



```
  GNU nano 2.2.6                    File: third.c                               Modified

#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int oslab = 0;

void *child(void *arg) {
    printf("oslab: %d, In child thread before changing the variable...\n", oslab);
    oslab = 2;
    printf("oslab: %d, In child thread after changing the variable...\n", oslab);
    return NULL;
}


int main() {
    printf("oslab: %d, In main thread before of anything...\n", oslab);
    pthread_t thread;
    oslab = 3;
    printf("oslab: %d, In main thread before creating child thread and after changing the variable.$
    pthread_create(&thread, NULL, child, NULL);
    pthread_join(thread, NULL);
    printf("oslab: %d, In main thread after child thread has finished...\n", oslab);
    return 0;
}
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

- [x] `FILL HERE with screenshot of output`



```
root@debian:~# gcc third.c -o third -lpthread
root@debian:~# ./third
oslab: 0, In main thread before of anything...
oslab: 3, In main thread before creating child thread and after changing the variable...
oslab: 3, In child thread before changing the variable...
oslab: 2, In child thread after changing the variable...
oslab: 2, In main thread after child thread has finished...
root@debian:~#
```

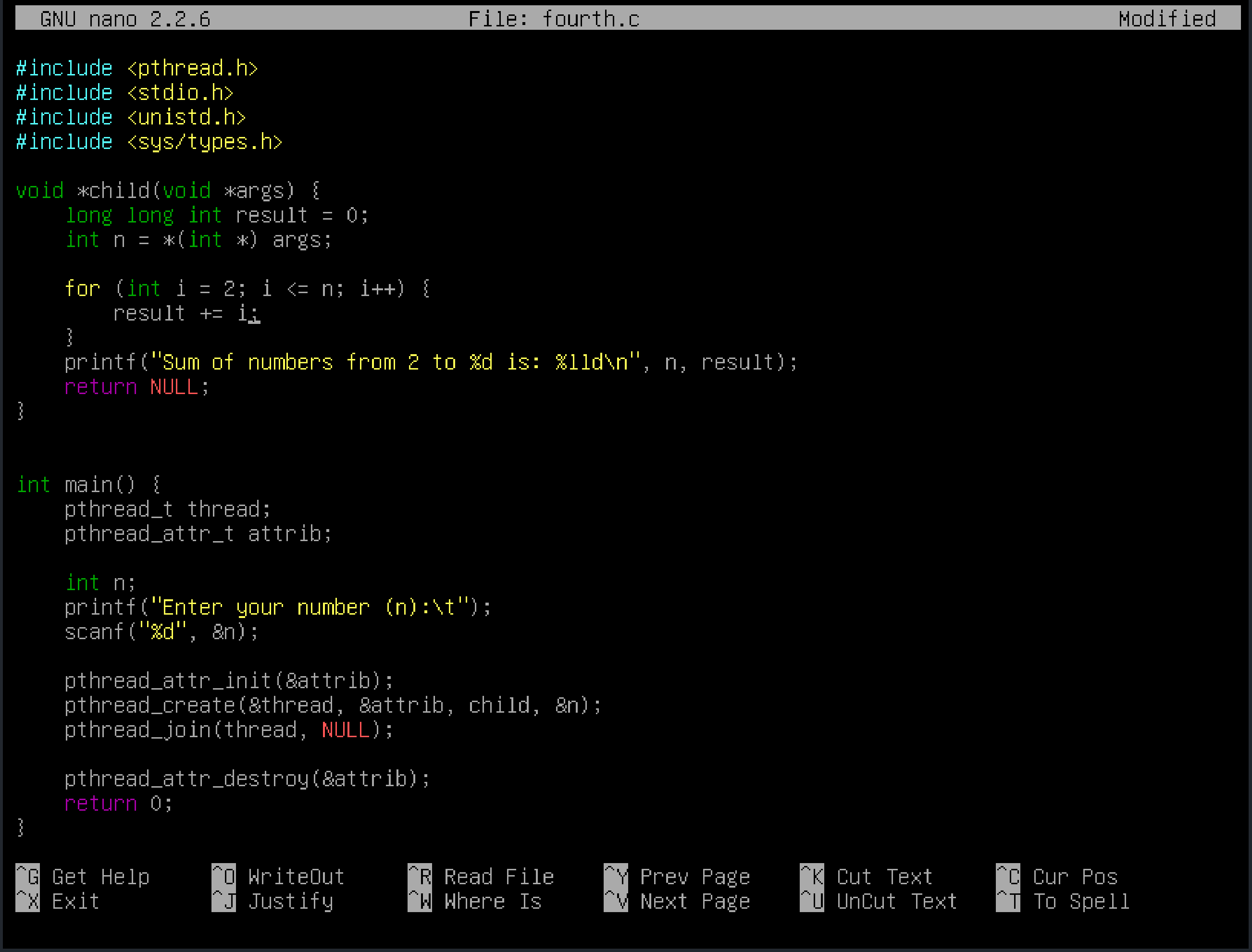
- [x] `FILL HERE with your descriptions (write in English or Persian) on how the variable has been changed and why`

خیر ریسه‌ها کپی جداگانه از متغیرها ندارند. متغیر سراسری در فضای حافظه مشترک قرار دارد بنابراین هر تغییری در این متغیرهای سراسری توسط هر کدام از ریسه‌های موجود یک پردازه، توسط سایر ریسه‌های آن نیز قابل مشاهده است. در همین بخش تغییراتی که ریسه فرزند در متغیر سراسری تعریف شده، اعمال نمود، در ریسه اصلی نیز قابل مشاهده و تاثیرگذار بود.

- [x] Sum of 2 to n

  - [x] `[FILL HERE with screenshot of code]`

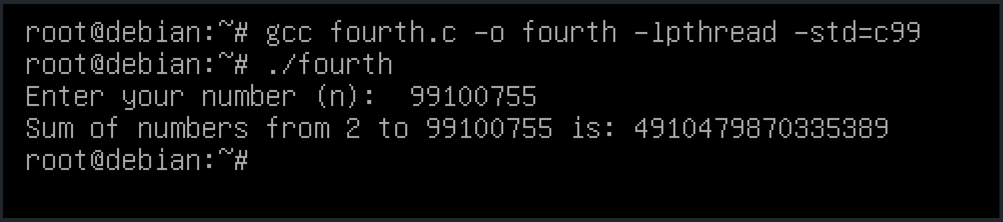
```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

void *child(void *args) {
    long long int result = 0;
    int n = *(int *) args;

    for (int i = 2; i <= n; i++) {
        result += i;
    }
    printf("Sum of numbers from 2 to %d is: %lld\n", n, result);
    return NULL;
}

int main() {
    pthread_t thread;
    pthread_attr_t attrib;
    int n;
    printf("Enter your number (n):\t");
    scanf("%d", &n);

    pthread_attr_init(&attrib);
    pthread_create(&thread, &attrib, child, &n);
    pthread_join(thread, NULL);

    pthread_attr_destroy(&attrib);
    return 0;
}
```



```
  GNU nano 2.2.6                  File: fourth.c                           Modified

#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

void *child(void *args) {
    long long int result = 0;
    int n = *(int *) args;

    for (int i = 2; i <= n; i++) {
        result += i;
    }
    printf("Sum of numbers from 2 to %d is: %lld\n", n, result);
    return NULL;
}


int main() {
    pthread_t thread;
    pthread_attr_t attrib;

    int n;
    printf("Enter your number (n):\t");
    scanf("%d", &n);

    pthread_attr_init(&attrib);
    pthread_create(&thread, &attrib, child, &n);
    pthread_join(thread, NULL);

    pthread_attr_destroy(&attrib);
    return 0;
}

^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

- [x] `[FILL HERE with screenshot of an execution of the code for n=(mean of team's student numbers)]`

  n = (99101087 + 99100422) / 2 = 99100755



```
root@debian:~# gcc fourth.c -o fourth -lpthread -std=c99
root@debian:~# ./fourth
Enter your number (n):   99100755
Sum of numbers from 2 to 99100755 is: 4910479870335389
root@debian:~#
```

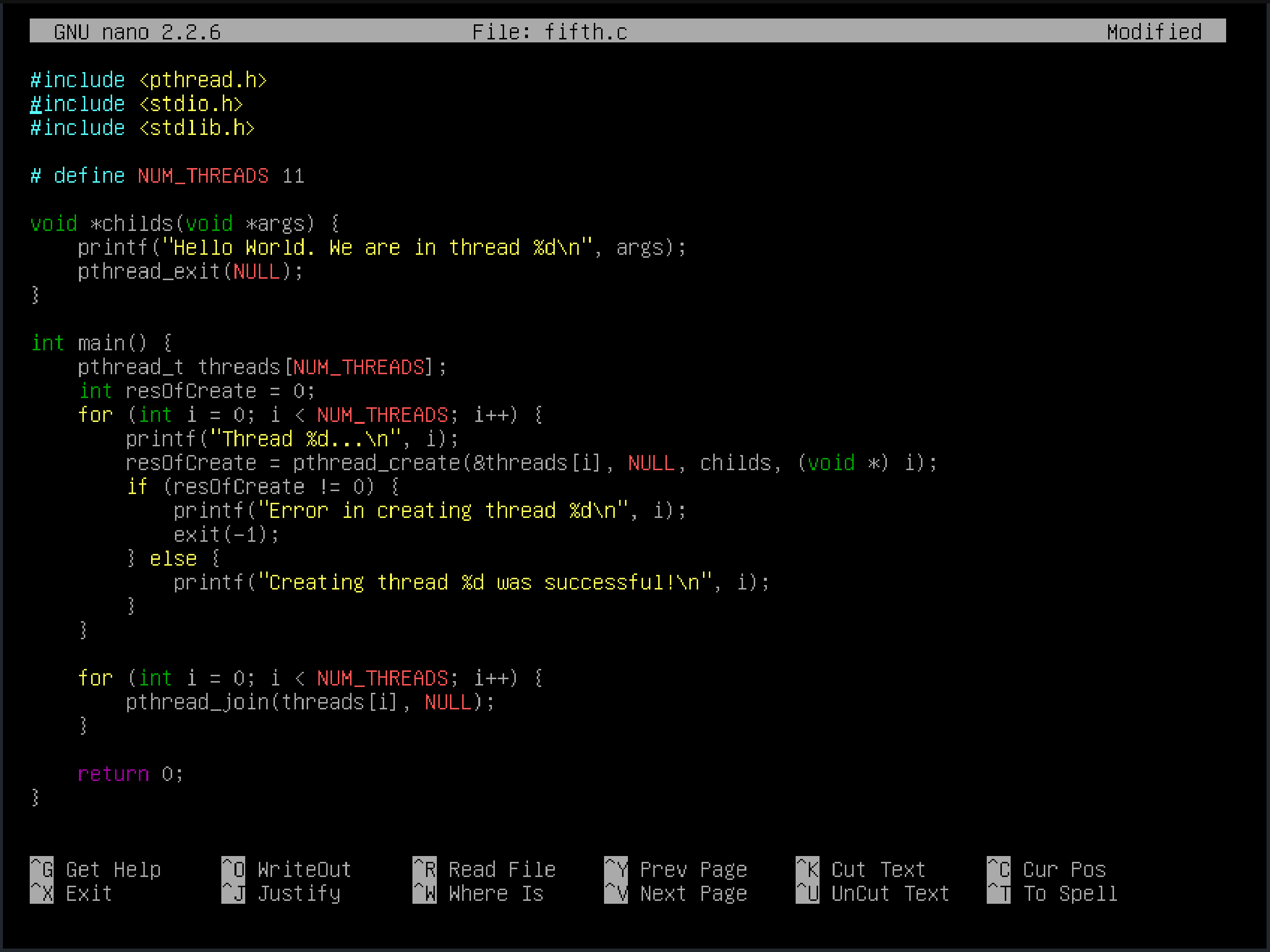
Section 7.4.2

- [x] Multiple threads

  - [x] `FILL HERE with screenshot of code`

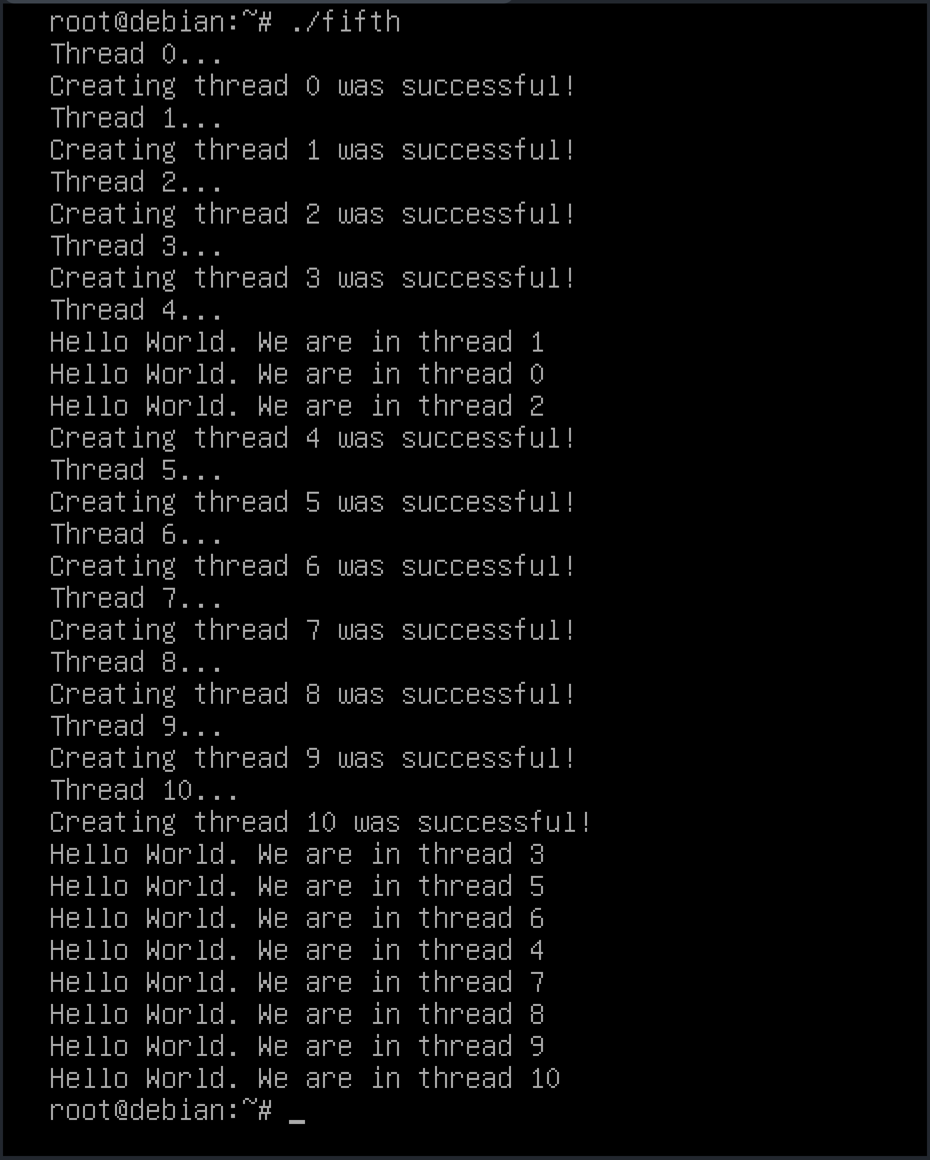
```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

# define NUM_THREADS 11
```

```c
void *childs(void *args) {
    printf("Hello World. We are in thread %d\n", args);
    pthread_exit(NULL);
}

int main() {
    pthread_t threads[NUM_THREADS];
    int resOfCreate = 0;

    for (int i = 0; i < NUM_THREADS; i++) {
        printf("Thread %d...\n", i);
        resOfCreate = pthread_create(&threads[i], NULL, childs, (void *) i);
        if (resOfCreate != 0) {
            printf("Error in creating thread %d\n", i);
            exit(-1);
        } else {
            printf("Creating thread %d was successful!\n", i);
        }
    }

    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    return 0;
}
```



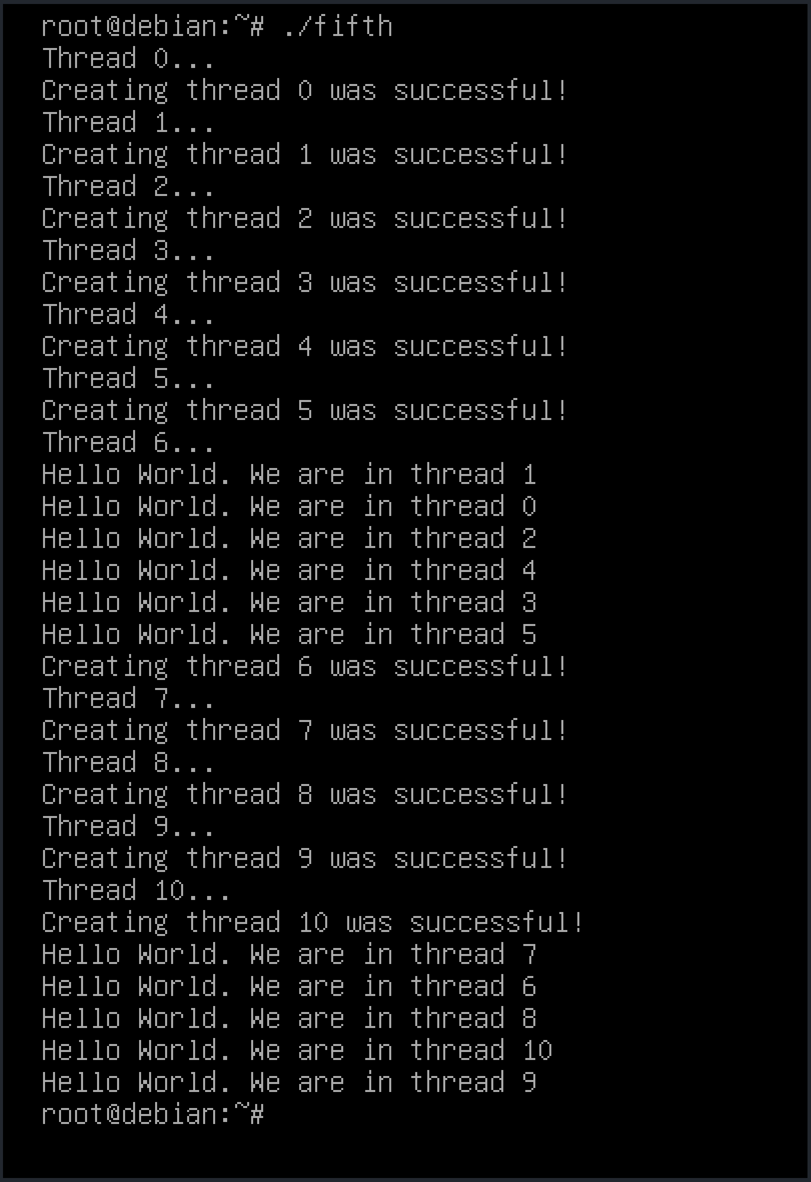
```
  GNU nano 2.2.6                  File: fifth.c                           Modified

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

# define NUM_THREADS 11

void *childs(void *args) {
    printf("Hello World. We are in thread %d\n", args);
    pthread_exit(NULL);
}

int main() {
    pthread_t threads[NUM_THREADS];
    int resOfCreate = 0;
    for (int i = 0; i < NUM_THREADS; i++) {
        printf("Thread %d...\n", i);
        resOfCreate = pthread_create(&threads[i], NULL, childs, (void *) i);
        if (resOfCreate != 0) {
            printf("Error in creating thread %d\n", i);
            exit(-1);
        } else {
            printf("Creating thread %d was successful!\n", i);
        }
    }

    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    return 0;
}


^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

- [x] `FILL HERE with execution of code`



```
root@debian:~# ./fifth
Thread 0...
Creating thread 0 was successful!
Thread 1...
Creating thread 1 was successful!
Thread 2...
Creating thread 2 was successful!
Thread 3...
Creating thread 3 was successful!
Thread 4...
Hello World. We are in thread 1
Hello World. We are in thread 0
Hello World. We are in thread 2
Creating thread 4 was successful!
Thread 5...
Creating thread 5 was successful!
Thread 6...
Creating thread 6 was successful!
Thread 7...
Creating thread 7 was successful!
Thread 8...
Creating thread 8 was successful!
Thread 9...
Creating thread 9 was successful!
Thread 10...
Creating thread 10 was successful!
Hello World. We are in thread 3
Hello World. We are in thread 5
Hello World. We are in thread 6
Hello World. We are in thread 4
Hello World. We are in thread 7
Hello World. We are in thread 8
Hello World. We are in thread 9
Hello World. We are in thread 10
root@debian:~# _
```

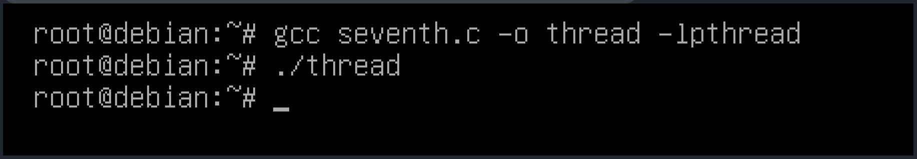
In another execution:



```
root@debian:~# ./fifth
Thread 0...
Creating thread 0 was successful!
Thread 1...
Creating thread 1 was successful!
Thread 2...
Creating thread 2 was successful!
Thread 3...
Creating thread 3 was successful!
Thread 4...
Creating thread 4 was successful!
Thread 5...
Creating thread 5 was successful!
Thread 6...
Hello World. We are in thread 1
Hello World. We are in thread 0
Hello World. We are in thread 2
Hello World. We are in thread 4
Hello World. We are in thread 3
Hello World. We are in thread 5
Creating thread 6 was successful!
Thread 7...
Creating thread 7 was successful!
Thread 8...
Creating thread 8 was successful!
Thread 9...
Creating thread 9 was successful!
Thread 10...
Creating thread 10 was successful!
Hello World. We are in thread 7
Hello World. We are in thread 6
Hello World. We are in thread 8
Hello World. We are in thread 10
Hello World. We are in thread 9
root@debian:~#
```

Section 7.4.3

- [x] Compiling the code
  - [x] `FILL HERE with screenshot of compilation`

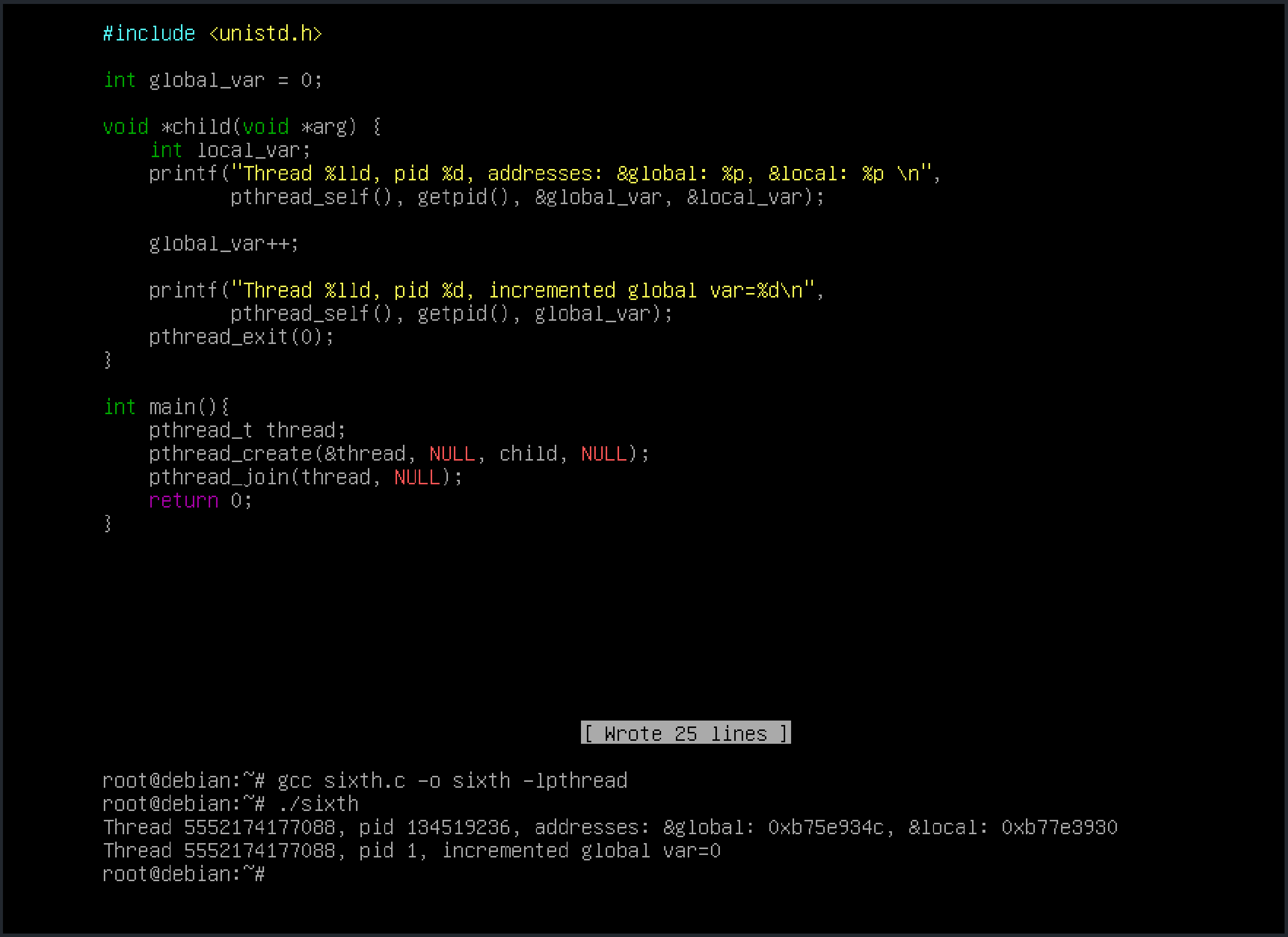


```
root@debian:~# gcc seventh.c -o thread -lpthread
root@debian:~# ./thread
root@debian:~# _
```

همچنین می‌توانیم مانند زیر عمل کنیم:

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
```

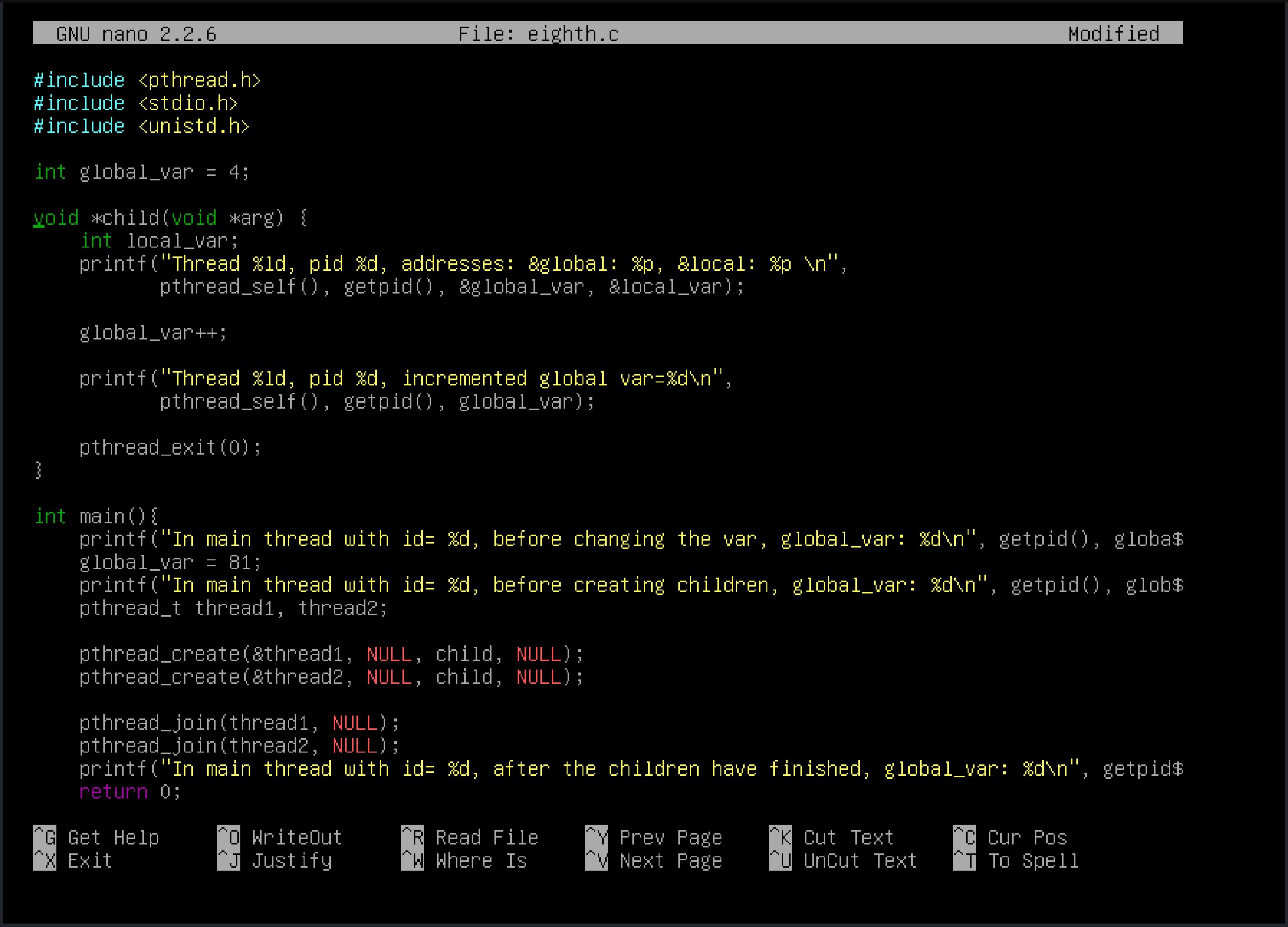
```
int global_var = 0;

void *child(void *arg) {
    int local_var;

    printf("Thread %ld, pid %d, addresses: &global: %p, &local: %p \n",
           pthread_self(), getpid(), &global_var, &local_var);

    global_var++;

    printf("Thread %ld, pid %d, incremented global var=%d\n",
           pthread_self(), getpid(), global_var);

    pthread_exit(0);
}

int main(){
    pthread_t thread;
    pthread_create(&thread, NULL, child, NULL);
    pthread_join(thread, NULL);
    return 0;
}
```



```
#include <unistd.h>

int global_var = 0;

void *child(void *arg) {
    int local_var;
    printf("Thread %lld, pid %d, addresses: &global: %p, &local: %p \n",
           pthread_self(), getpid(), &global_var, &local_var);

    global_var++;

    printf("Thread %lld, pid %d, incremented global var=%d\n",
           pthread_self(), getpid(), global_var);
    pthread_exit(0);
}

int main(){
    pthread_t thread;
    pthread_create(&thread, NULL, child, NULL);
    pthread_join(thread, NULL);
    return 0;
}

                              [ Wrote 25 lines ]

root@debian:~# gcc sixth.c -o sixth -lpthread
root@debian:~# ./sixth
Thread 5552174177088, pid 134519236, addresses: &global: 0xb75e934c, &local: 0xb77e3930
Thread 5552174177088, pid 1, incremented global var=0
root@debian:~#
```

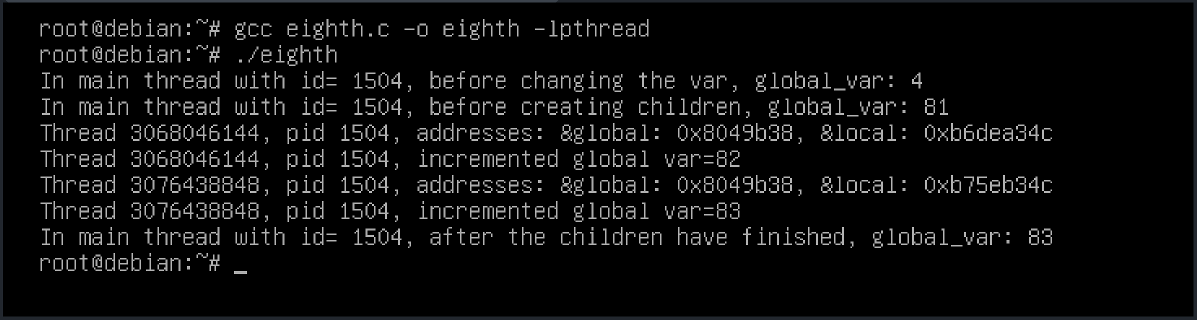
- [x] global_param
  - [x] `FILL HERE with screenshot of code`

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

int global_var = 4;

void *child(void *arg) {
    int local_var;

    printf("Thread %ld, pid %d, addresses: &global: %p, &local: %p \n",
           pthread_self(), getpid(), &global_var, &local_var);

    global_var++;

    printf("Thread %ld, pid %d, incremented global var=%d\n",
           pthread_self(), getpid(), global_var);

    pthread_exit(0);
}

int main(){
    printf("In main thread with id= %d, before changing the var, global_var:
%d\n", getpid(), global_var);
    global_var = 81;
    printf("In main thread with id= %d, before creating children, global_var:
%d\n", getpid(), global_var);
    pthread_t thread1, thread2;
    pthread_create(&thread1, NULL, child, NULL);
    pthread_create(&thread2, NULL, child, NULL);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    printf("In main thread with id= %d, after the children have finished,
global_var: %d\n", getpid(), global_var);
    return 0;
}
```



```
  GNU nano 2.2.6                 File: eighth.c                          Modified

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

int global_var = 4;

void *child(void *arg) {
    int local_var;
    printf("Thread %ld, pid %d, addresses: &global: %p, &local: %p \n",
           pthread_self(), getpid(), &global_var, &local_var);

    global_var++;

    printf("Thread %ld, pid %d, incremented global var=%d\n",
           pthread_self(), getpid(), global_var);

    pthread_exit(0);
}

int main(){
    printf("In main thread with id= %d, before changing the var, global_var: %d\n", getpid(), globa$
    global_var = 81;
    printf("In main thread with id= %d, before creating children, global_var: %d\n", getpid(), glob$
    pthread_t thread1, thread2;

    pthread_create(&thread1, NULL, child, NULL);
    pthread_create(&thread2, NULL, child, NULL);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    printf("In main thread with id= %d, after the children have finished, global_var: %d\n", getpid$
    return 0;

^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

- [x] `FILL HERE with screenshot of output`



```
root@debian:~# gcc eighth.c -o eighth -lpthread
root@debian:~# ./eighth
In main thread with id= 1504, before changing the var, global_var: 4
In main thread with id= 1504, before creating children, global_var: 81
Thread 3068046144, pid 1504, addresses: &global: 0x8049b38, &local: 0xb6dea34c
Thread 3068046144, pid 1504, incremented global var=82
Thread 3076438848, pid 1504, addresses: &global: 0x8049b38, &local: 0xb75eb34c
Thread 3076438848, pid 1504, incremented global var=83
In main thread with id= 1504, after the children have finished, global_var: 83
root@debian:~# _
```

همانطور که مشاهده می‌کنید، در ابتدا مقدار متغیر برابر مقدار اولیه است. پس از اینکه ریسه اصلی آن را مقدار دهی می‌کند، این متغیر مقدار جدید را می‌پذیرد. سپس دو ریسه فرزند ایجاد می‌کنیم. شماره ریسه‌ها قابل ملاحظه است. همانطور که می‌بینید آدرس متغیر عمومی برای هر دو ریسه یکسان است زیرا از حافظه مشترک و یکسان پردازه بهره می‌برند. تمامی شماره پردازه‌ها همانطور که انتظار داشتیم، یکسان است. تغییرهایی که یک ریسه در متغیر عمومی ایجاد کرده نیز توسط بقیه ریسه‌ها قابل مشاهده و تاثیرگذار است.

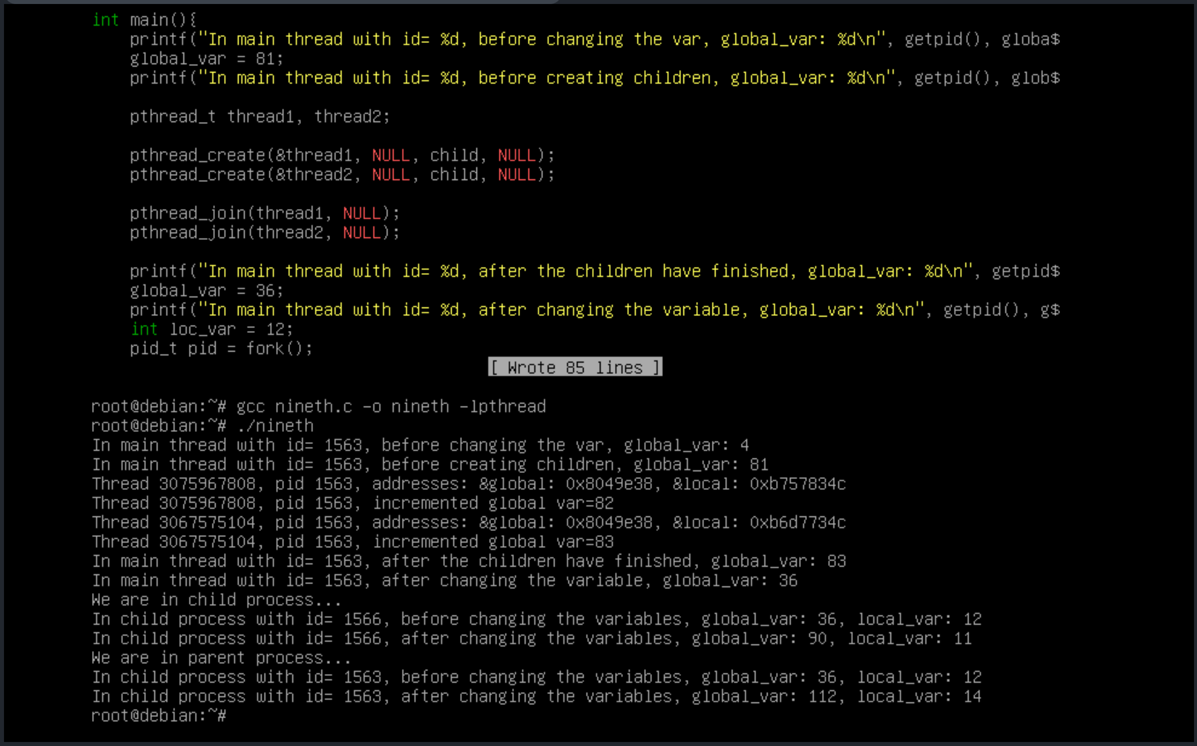
- [x] Forking

- [x] `FILL HERE with screenshot of code`

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
```

```
#include <sys/types.h>

int global_var = 4;

void *child(void *arg) {
    int local_var;

    printf("Thread %lu, pid %d, addresses: &global: %p, &local: %p \n",
           pthread_self(), getpid(), &global_var, &local_var);

    global_var++;

    printf("Thread %lu, pid %d, incremented global var=%d\n",
           pthread_self(), getpid(), global_var);

    pthread_exit(0);
}

int main(){
    printf("In main thread with id= %d, before changing the var, global_var:
%d\n", getpid(), global_var);
    global_var = 81;
    printf("In main thread with id= %d, before creating children, global_var:
%d\n", getpid(), global_var);
    pthread_t thread1, thread2;
    pthread_create(&thread1, NULL, child, NULL);
    pthread_create(&thread2, NULL, child, NULL);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    printf("In main thread with id= %d, after the children have finished,
global_var: %d\n", getpid(), global_var);

    global_var = 36;
    printf("In main thread with id= %d, after changing the variable, global_var:
%d\n", getpid(), global_var);
    int loc_var = 12;
    pid_t pid = fork();

    if (pid<0){
        printf("Error!");
        return 1;
    } else if (pid == 0){
        printf("We are in child process...\n");
        printf("In child process with id= %d, before changing the variables,
global_var: %d, local_var: %d\n",
               getpid(), global_var, loc_var);
        global_var = 90;
        loc_var = 11;
        printf("In child process with id= %d, after changing the variables,
global_var: %d, local_var: %d\n",
               getpid(), global_var, loc_var);
    } else {
        wait(NULL);
        printf("We are in parent process...\n");
        printf("In child process with id= %d, before changing the variables,
global_var: %d, local_var: %d\n",
               getpid(), global_var, loc_var);
        global_var = 112;
        loc_var = 14;
        printf("In child process with id= %d, after changing the variables,
global_var: %d, local_var: %d\n",
               getpid(), global_var, loc_var);
    }
    return 0;
}
```

- [x] `FILL HERE with screenshot of output`



```
int main(){
    printf("In main thread with id= %d, before changing the var, global_var: %d\n", getpid(), globa$
    global_var = 81;
    printf("In main thread with id= %d, before creating children, global_var: %d\n", getpid(), glob$

    pthread_t thread1, thread2;

    pthread_create(&thread1, NULL, child, NULL);
    pthread_create(&thread2, NULL, child, NULL);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    printf("In main thread with id= %d, after the children have finished, global_var: %d\n", getpid$
    global_var = 36;
    printf("In main thread with id= %d, after changing the variable, global_var: %d\n", getpid(), g$
    int loc_var = 12;
    pid_t pid = fork();
                                        [ Wrote 85 lines ]

root@debian:~# gcc nineth.c -o nineth -lpthread
root@debian:~# ./nineth
In main thread with id= 1563, before changing the var, global_var: 4
In main thread with id= 1563, before creating children, global_var: 81
Thread 3075967808, pid 1563, addresses: &global: 0x8049e38, &local: 0xb757834c
Thread 3075967808, pid 1563, incremented global var=82
Thread 3067575104, pid 1563, addresses: &global: 0x8049e38, &local: 0xb6d7734c
Thread 3067575104, pid 1563, incremented global var=83
In main thread with id= 1563, after the children have finished, global_var: 83
In main thread with id= 1563, after changing the variable, global_var: 36
We are in child process...
In child process with id= 1566, before changing the variables, global_var: 36, local_var: 12
In child process with id= 1566, after changing the variables, global_var: 90, local_var: 11
We are in parent process...
In child process with id= 1563, before changing the variables, global_var: 36, local_var: 12
In child process with id= 1563, after changing the variables, global_var: 112, local_var: 14
root@debian:~#
```

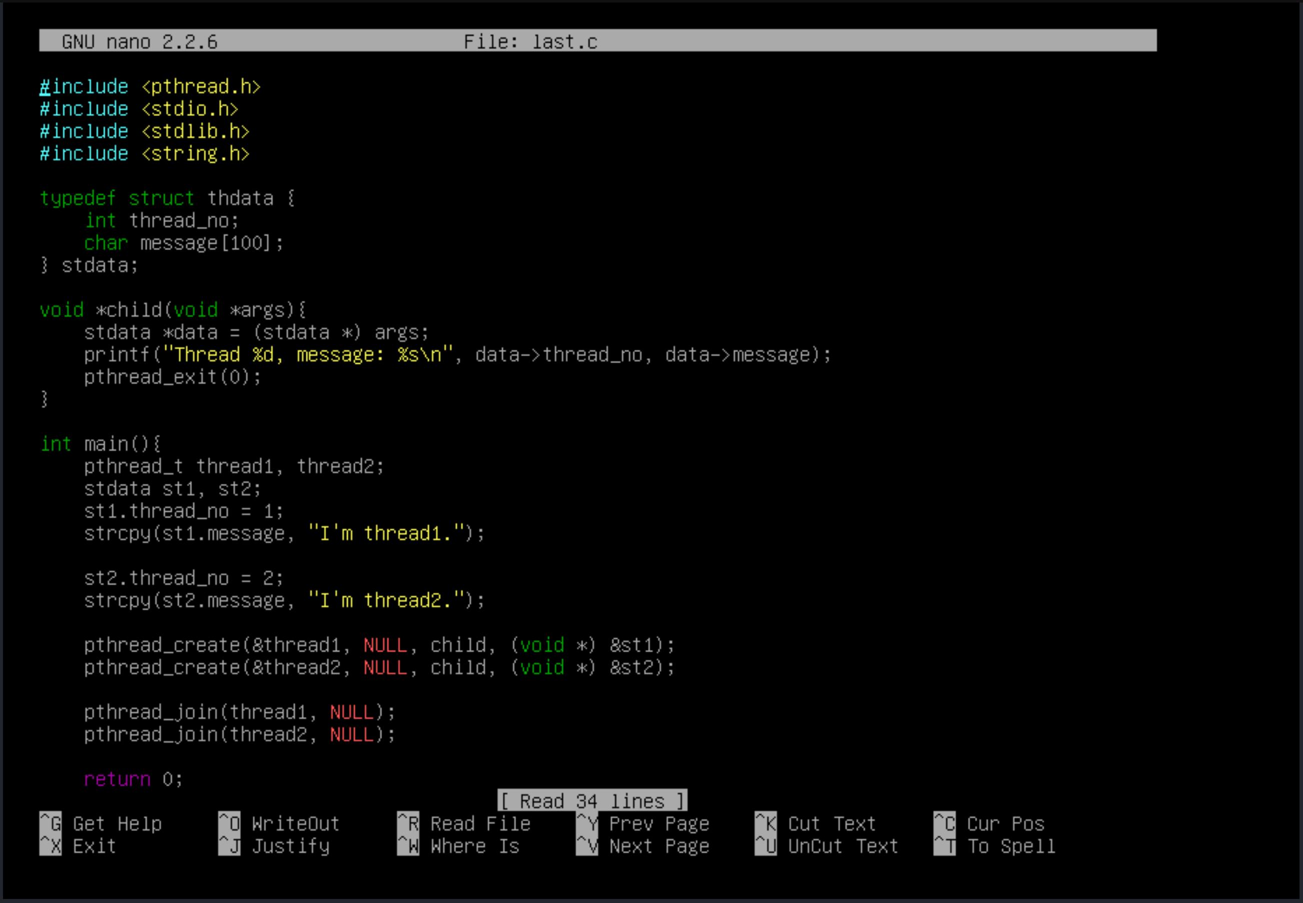
در این بخش در ادامه بخش قبلی، بعد انجام تغییرات ریسه‌ها، به سراغ فورک و ساخت پردازه جدید خواهیم رفت. همانطور که مشاهده می‌کنید، بعد اینکه پردازه فرزند کارش تمام شد، هنگامی که پردازه والد شروع به کار می‌کند، مقادیری که پردازه فرزند تغییر داده است را نمی‌بینند.

Section 7.4.4

- [x] Passing multiple variables

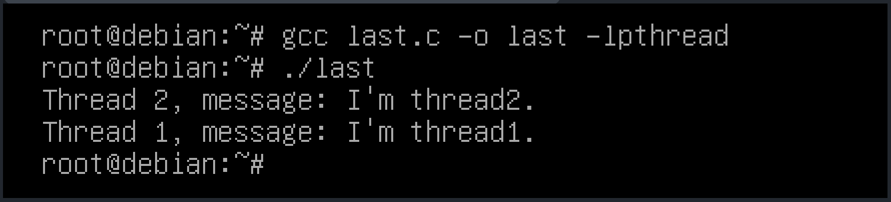
  - [x] `FILL HERE with screenshot of code`

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

typedef struct thdata {
    int thread_no;
    char message[100];
} stdata;

void *child(void *args){
    stdata *data = (stdata *) args;
    printf("Thread %d, message: %s\n", data->thread_no, data->message);
    pthread_exit(0);
}

int main(){
    pthread_t thread1, thread2;
    stdata st1, st2;
    st1.thread_no = 1;
    strcpy(st1.message, "I'm thread1.");

    st2.thread_no = 2;
    strcpy(st2.message, "I'm thread2.");

    pthread_create(&thread1, NULL, child, (void *) &st1);
    pthread_create(&thread2, NULL, child, (void *) &st2);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    return 0;
}
```



```
  GNU nano 2.2.6                  File: last.c

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

typedef struct thdata {
    int thread_no;
    char message[100];
} stdata;

void *child(void *args){
    stdata *data = (stdata *) args;
    printf("Thread %d, message: %s\n", data->thread_no, data->message);
    pthread_exit(0);
}

int main(){
    pthread_t thread1, thread2;
    stdata st1, st2;
    st1.thread_no = 1;
    strcpy(st1.message, "I'm thread1.");

    st2.thread_no = 2;
    strcpy(st2.message, "I'm thread2.");

    pthread_create(&thread1, NULL, child, (void *) &st1);
    pthread_create(&thread2, NULL, child, (void *) &st2);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    return 0;
                                         [ Read 34 lines ]
^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

- [x] `FILL HERE with screenshot of output`



```
root@debian:~# gcc last.c -o last -lpthread
root@debian:~# ./last
Thread 2, message: I'm thread2.
Thread 1, message: I'm thread1.
root@debian:~#
```

Amirreza81 added the documentation label 2 weeks ago

Amirreza81 assigned **AMshoka** 2 weeks ago

Assignees

AMshoka

Labels

documentation

Projects

None yet

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.

2 participants

Pin issue