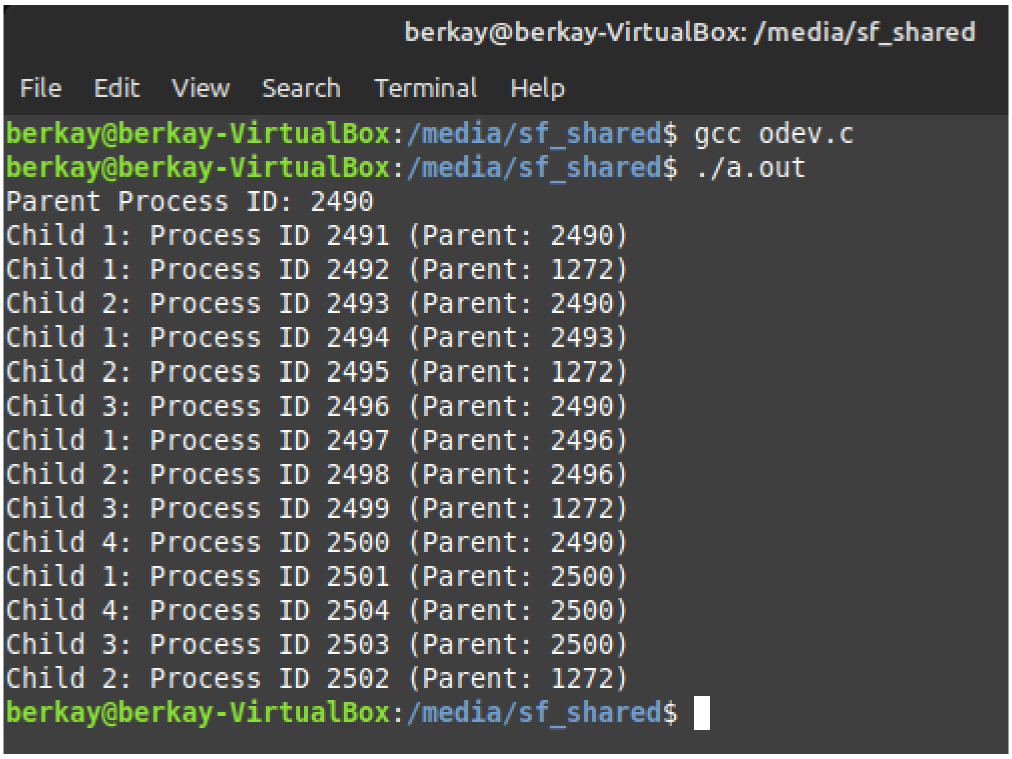
BLG 312E

COMPUTER OPERATING SYSTEMS

ASSIGNMENT 1

BERKAY SANCI 504231561

1.

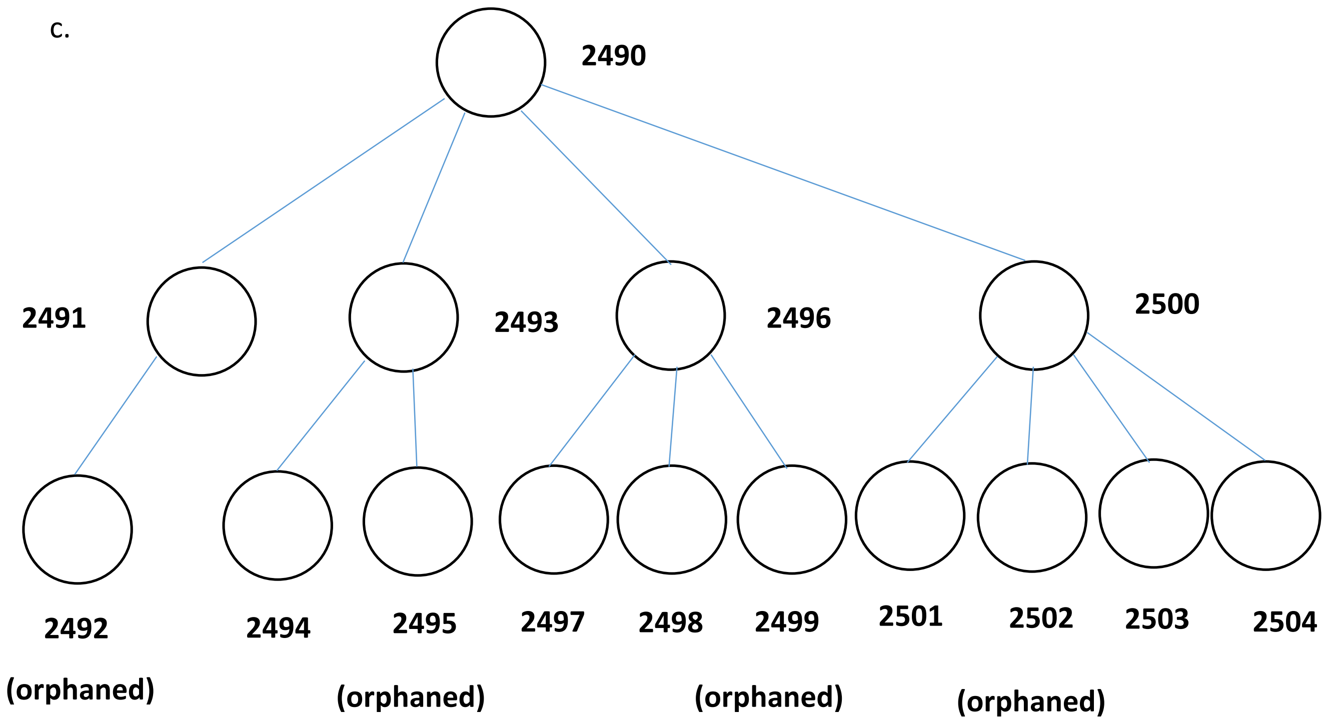

berkay@berkay-VirtualBox: /media/sf_shared
File Edit View Search Terminal Help

```
berkay@berkay-VirtualBox:/media/sf_shared$ gcc odev.c
berkay@berkay-VirtualBox:/media/sf_shared$ ./a.out
Parent Process ID: 2490
Child 1: Process ID 2491 (Parent: 2490)
Child 1: Process ID 2492 (Parent: 1272)
Child 2: Process ID 2493 (Parent: 2490)
Child 1: Process ID 2494 (Parent: 2493)
Child 2: Process ID 2495 (Parent: 1272)
Child 3: Process ID 2496 (Parent: 2490)
Child 1: Process ID 2497 (Parent: 2496)
Child 2: Process ID 2498 (Parent: 2496)
Child 3: Process ID 2499 (Parent: 1272)
Child 4: Process ID 2500 (Parent: 2490)
Child 1: Process ID 2501 (Parent: 2500)
Child 4: Process ID 2504 (Parent: 2500)
Child 3: Process ID 2503 (Parent: 2500)
Child 2: Process ID 2502 (Parent: 1272)
berkay@berkay-VirtualBox:/media/sf_shared$ 
```

a. System call fork() is called 14 times. It is called 4 times for the outer loop. For inner loop, it is called 1,2,3,4 times respectively for each iteration of the outer loop. So, in total, it is called 4+1+2+3+4=14 times.

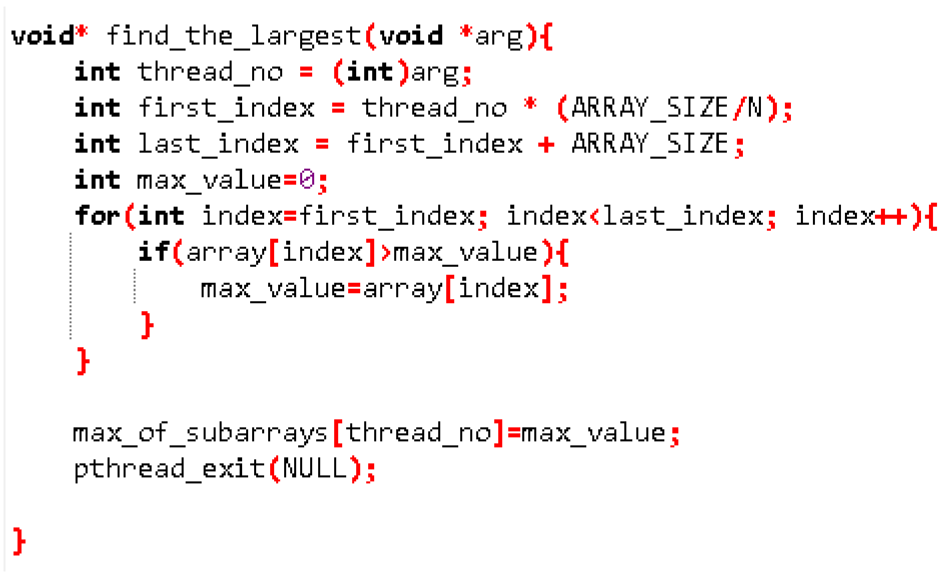
b. The program creates 15 processes in total, including the parent process. The original parent process has 4 children. And children has 1,2,3,4 children respectively. Fort he inner loop, since parents do not wait for children to execute, some children becomes orphan as it is seen on the output given above. Including the original parent process, 15 processes are created.

c.



d. exit() system is called to terminate a process. Also, all the resources of the process are deallocated and reclaimed by the operating system with exit() call. For the fork process, parent process can terminate its children with this call. Its absence might result in resource leaks and undesired executing of children processes.

2. For given task, I first created the function that each thread is going to use.



```
void* find_the_largest(void *arg){
    int thread_no = (int)arg;
    int first_index = thread_no * (ARRAY_SIZE/N);
    int last_index = first_index + ARRAY_SIZE;
    int max_value=0;
    for(int index=first_index; index<last_index; index++){
        if(array[index]>max_value){
            max_value=array[index];
        }
    }

    max_of_subarrays[thread_no]=max_value;
    pthread_exit(NULL);

}
```

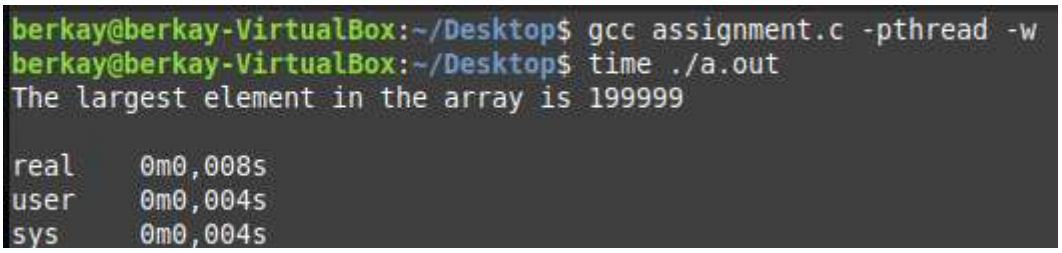
I divided the main array into sub-arrays and assigned the first and last index for each thread based on the thread number. Then, for each thread, largest element in the corresponding sub-array is found.

```
int main(int argc, char *argv[]){
    int i,t;
    pthread_t threads[N];
    int max_all = 0;


    for(i=0;i<ARRAY_SIZE;i++){
        array[i]=i;
    }

    for(t=0;t<N;t++){
        pthread_create(&threads[t], NULL, find_the_largest, (void*)t);
    }

    for(t=0;t<N;t++){
        pthread_join(threads[t],NULL);
    }

    for(t=0;t<N;t++){
        if(max_of_subarrays[t]>max_all){
            max_all = max_of_subarrays[t];
        }
    }

    printf("\nThe largest element in the array is %d\n",max_all);
    pthread_exit(NULL);
}
```

Then, in the main function, I initialized the main array. After that, I created N number of threads using pthread_create() function where N is defined as different numbers for the task. Then threads are joined using pthread_join() function and at the and maximum element of the array is determined by comparing maximum elements of all the threads.

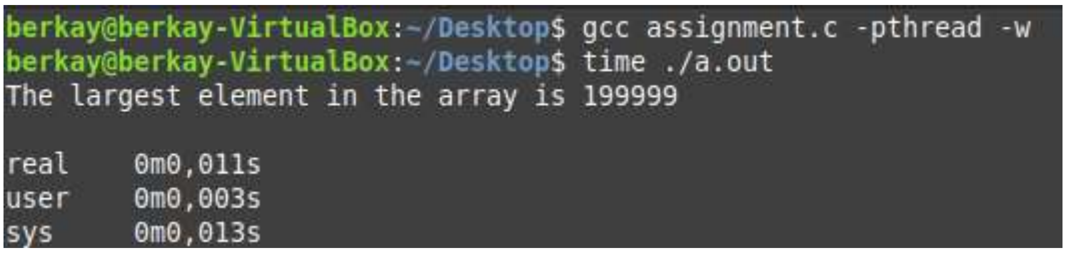
For time measuring application, I used an array with size of 200000 and defined N (number of threads) differently on each run.

N=1:



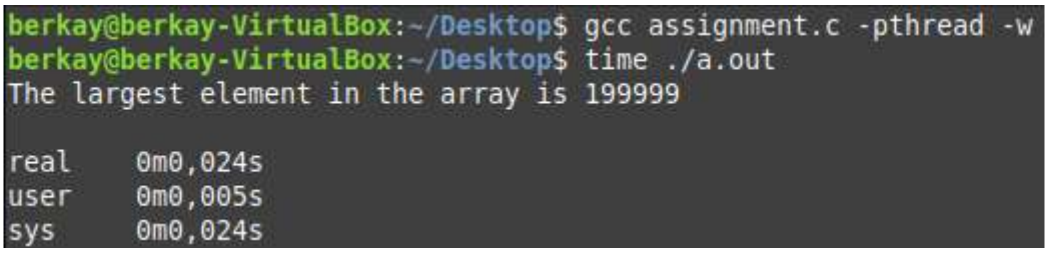
```
berkay@berkay-VirtualBox:~/Desktop$ gcc assignment.c -pthread -w
berkay@berkay-VirtualBox:~/Desktop$ time ./a.out
The largest element in the array is 199999

real    0m0,008s
user    0m0,004s
sys     0m0,004s
```

N=10:



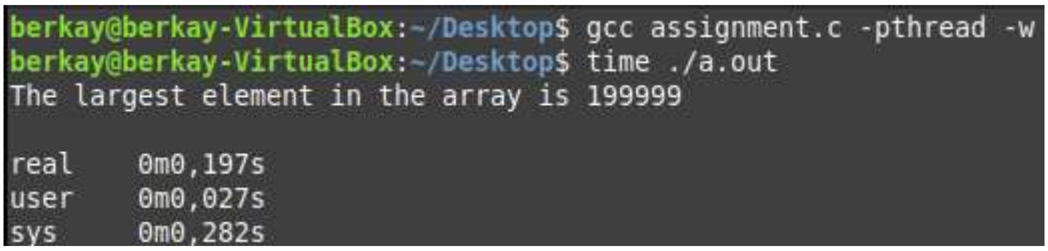
```
berkay@berkay-VirtualBox:~/Desktop$ gcc assignment.c -pthread -w
berkay@berkay-VirtualBox:~/Desktop$ time ./a.out
The largest element in the array is 199999

real    0m0,011s
user    0m0,003s
sys     0m0,013s
```

N=100:



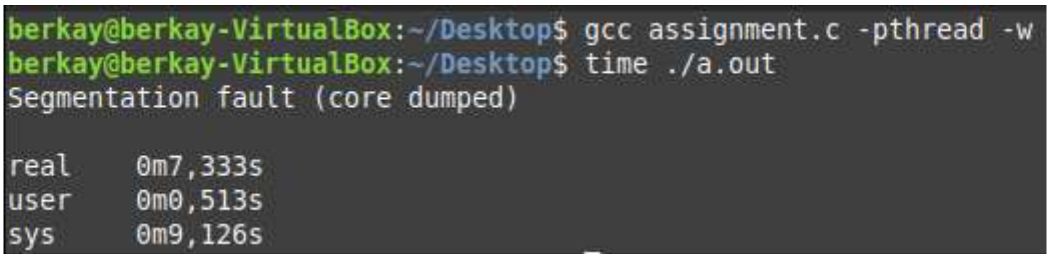
```
berkay@berkay-VirtualBox:~/Desktop$ gcc assignment.c -pthread -w
berkay@berkay-VirtualBox:~/Desktop$ time ./a.out
The largest element in the array is 199999

real    0m0,024s
user    0m0,005s
sys     0m0,024s
```

N=1000:



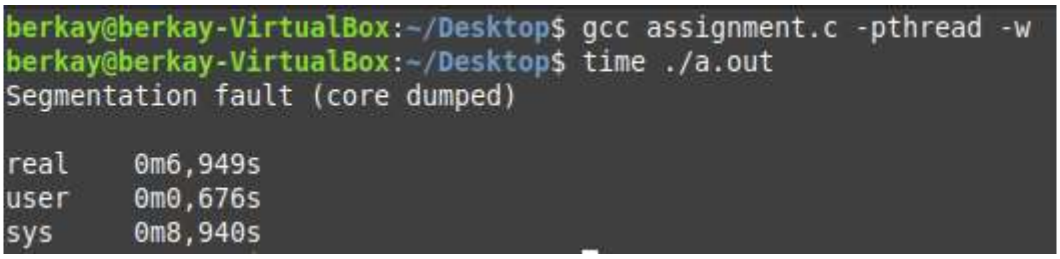
```
berkay@berkay-VirtualBox:~/Desktop$ gcc assignment.c -pthread -w
berkay@berkay-VirtualBox:~/Desktop$ time ./a.out
The largest element in the array is 199999

real    0m0,197s
user    0m0,027s
sys     0m0,282s
```

N=100000:



```
berkay@berkay-VirtualBox:~/Desktop$ gcc assignment.c -pthread -w
berkay@berkay-VirtualBox:~/Desktop$ time ./a.out
Segmentation fault (core dumped)

real    0m7,333s
user    0m0,513s
sys     0m9,126s
```

N=200000:



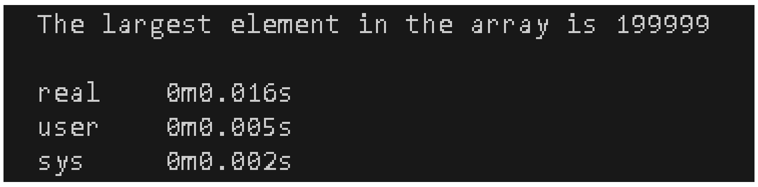
```
berkay@berkay-VirtualBox:~/Desktop$ gcc assignment.c -pthread -w
berkay@berkay-VirtualBox:~/Desktop$ time ./a.out
Segmentation fault (core dumped)

real    0m6,949s
user    0m0,676s
sys     0m8,940s
```

From N=1 to N=1000, the actual time elapsed and the number of CPU seconds in kernel mode increases. However, user time varies over increasing number of threads.

For N=100000 and N=200000, I get segmentation fault (core dumped) error. This is because I extend the limit of threads that is allowed to create. When I use ‘sysctl kernel.threads-max’ command, it gives me an output around 35000. I tried to increase the limit but I kept getting segmentation faults. So I did not go any further.

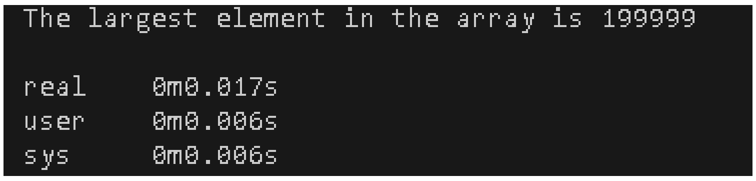
Later, I decided to try it on ITU SSH and it worked for N=100000 and N=200000. So I compiled an ran it for all N values on ITU SSH.

N=1:



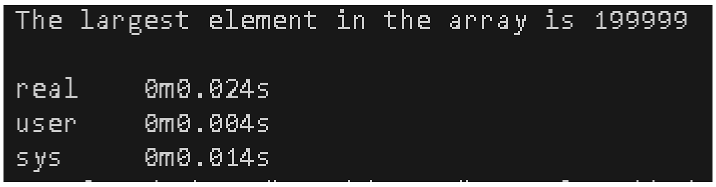
```
The largest element in the array is 199999

real    0m0.016s
user    0m0.005s
sys     0m0.002s
```

N=10:



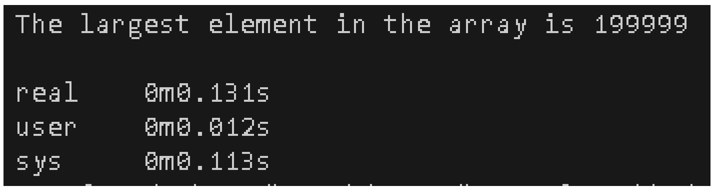
```
The largest element in the array is 199999

real    0m0.017s
user    0m0.006s
sys     0m0.006s
```

N=100:



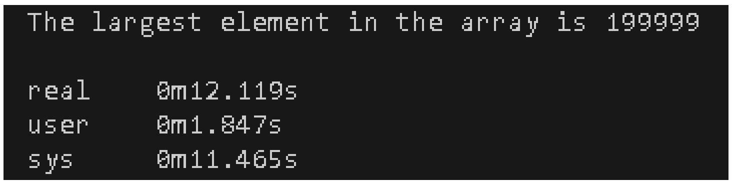
```
The largest element in the array is 199999

real    0m0.024s
user    0m0.004s
sys     0m0.014s
```

N=1000:



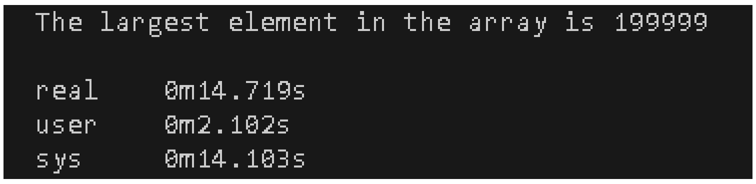
```
The largest element in the array is 199999

real    0m0.131s
user    0m0.012s
sys     0m0.113s
```

N=100000:

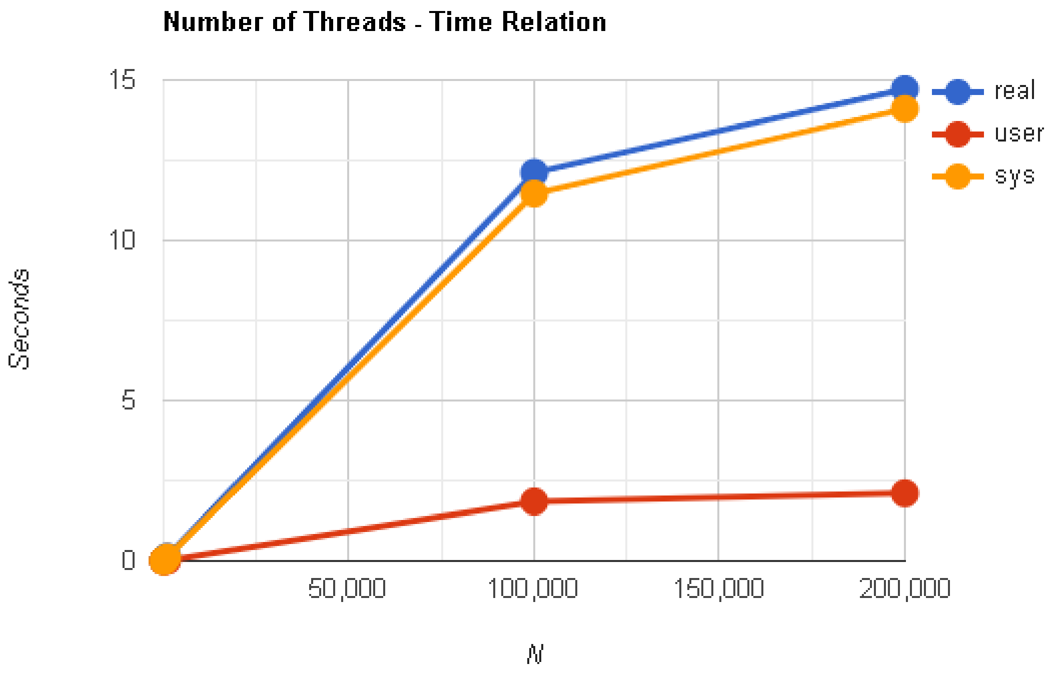


```
The largest element in the array is 199999

real	0m12.119s
user	0m1.847s
sys	0m11.465s
```

N=200000:



```
The largest element in the array is 199999

real	0m14.719s
user	0m2.102s
sys	0m14.103s
```



With the findings given above, this graph is obtained.