HPC
Experiment 4
Quicksort (Serial and Parallel)

Hrushikesh Pandit
63 TYCSE
Panel F

<u>Code:</u>
Quicksort (Serial):

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

static int size_arr;
static long *iarr;
static void quicksort(int, int);
static int partition(int, int);
static void swap(long*, long*);

int main(void) {
   int i = 0;

   srand(time(NULL));
   (void)printf("What is the size of array?");
   (void)scanf("%d", &size_arr);

   iarr = (long*) malloc(size_arr * sizeof(long int));
   if(iarr == NULL) {
      exit(EXIT_FAILURE);
   }

   *iarr = 0;


   (void)printf("Array Input: ");

   while(i < size_arr) {
      // iarr[i] = rand() % 100;
      *(iarr+i++) = rand();
   }

   quicksort(0, size_arr-1);

   (void)printf("Array Sorted: ");
```

```
    i = 0;
    while(i < size_arr) {
        (void)printf("%ld ", *(iarr+i++));
    }

    (void)printf("\n");
    free(iarr);

    return 0;
}

void swap(long *a, long *b) {
    long temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int low, int high) {
    long pivot = iarr[low];
    int leftwall = (int)low;
    int i;

    for(i = low + 1; i <= high; i++) {
        if(pivot > iarr[i]) {
            swap(&iarr[i], &iarr[leftwall]);
            ++leftwall;
        }
    }

    swap(&iarr[leftwall], &pivot);

    return leftwall;
}

void quicksort(int low, int high) {
    if(low < high) {
        int pivot = partition(low, high);
        quicksort(low, pivot);
        quicksort(pivot+1, high);
    }
}
```

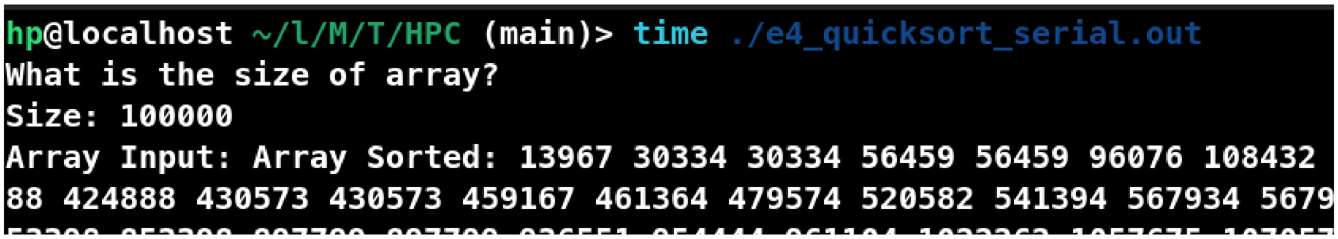
Quicksort (Parallel):

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
static int size_arr;
static long *iarr;
static void quicksort(int, int);
static int partition(int, int);
static void swap(long*, long*);

int main(void) {
  int i = 0;
  srand(time(NULL));
  (void)printf("What is the size of array?");
  (void)scanf("%d", &size_arr);

  iarr = (long*) malloc(size_arr * sizeof(long int));
  if(iarr == NULL) {
    exit(EXIT_FAILURE);
  }

  *iarr = 0;


  (void)printf("Array Input: ");
  #pragma omp parallel num_threads(100)
  {
    while(i < size_arr) {
      *(iarr+i++) = rand();
    }
  }

  quicksort(0, size_arr-1);

  (void)printf("Array Sorted: ");
  i = 0;
  while(i < size_arr) {
    (void)printf("%ld ", *(iarr+i++));
  }

  (void)printf("\n");
  free(iarr);
```

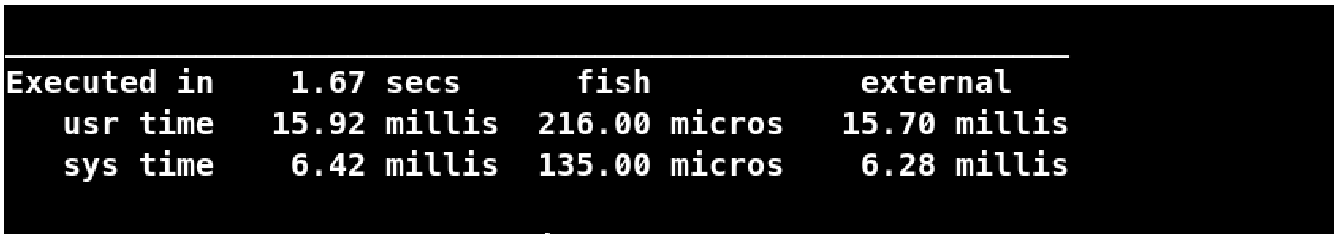
```
    return 0;
}

void swap(long *a, long *b) {
    long temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int low, int high) {
    long pivot = iarr[low];
    int leftwall = (int)low;
    int i;

    for(i = low + 1; i <= high; i++) {
        if(pivot > iarr[i]) {
            swap(&iarr[i], &iarr[leftwall]);
            ++leftwall;
        }
    }

    swap(&iarr[leftwall], &pivot);

    return leftwall;
}

void quicksort(int low, int high) {
    if(low < high) {
        int pivot = partition(low, high);

        #pragma omp parallel sections
        {
            #pragma omp section
            {
                quicksort(low, pivot);
            }
            #pragma omp section
            {
                quicksort(pivot+1, high);
            }
        }
    }
}
```

Output:

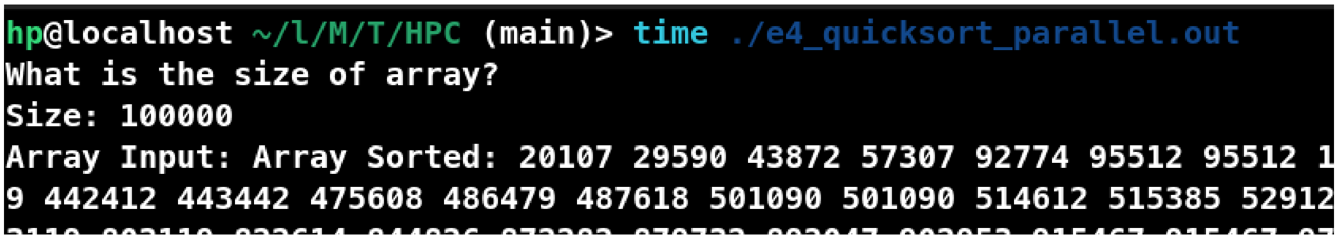
100k records:



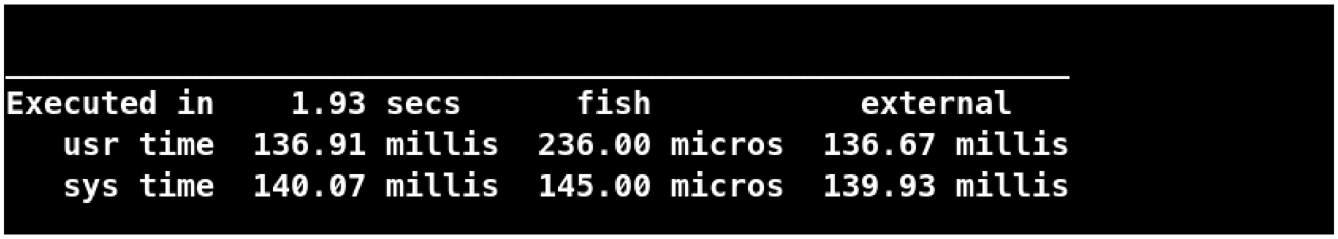
```
hp@localhost ~/l/M/T/HPC (main)> time ./e4_quicksort_serial.out
What is the size of array?
Size: 100000
Array Input: Array Sorted: 13967 30334 30334 56459 56459 96076 108432
88 424888 430573 430573 459167 461364 479574 520582 541394 567934 5679
```



```
________________________________________________________
Executed in    1.67 secs      fish           external
   usr time   15.92 millis  216.00 micros   15.70 millis
   sys time    6.42 millis  135.00 micros    6.28 millis
```

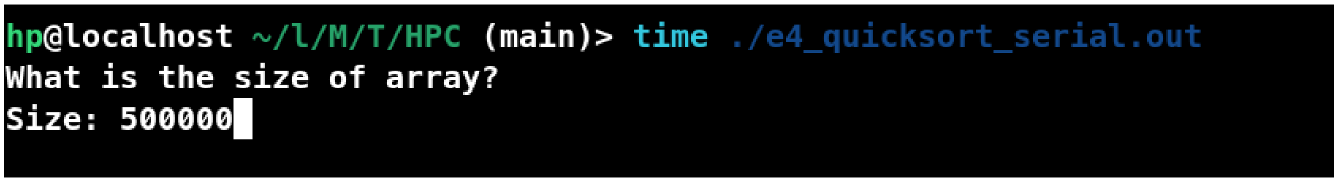


```
hp@localhost ~/l/M/T/HPC (main)> time ./e4_quicksort_parallel.out
What is the size of array?
Size: 100000
Array Input: Array Sorted: 20107 29590 43872 57307 92774 95512 95512 1
9 442412 443442 475608 486479 487618 501090 501090 514612 515385 52912
```

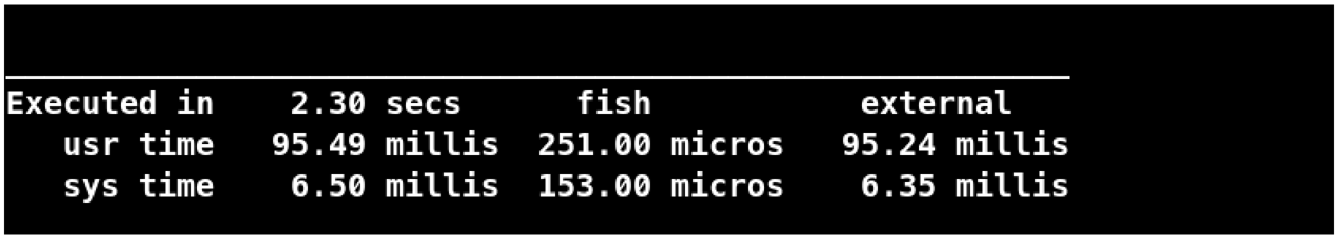


```
________________________________________________________
Executed in    1.93 secs      fish           external
   usr time  136.91 millis  236.00 micros  136.67 millis
   sys time  140.07 millis  145.00 micros  139.93 millis
```

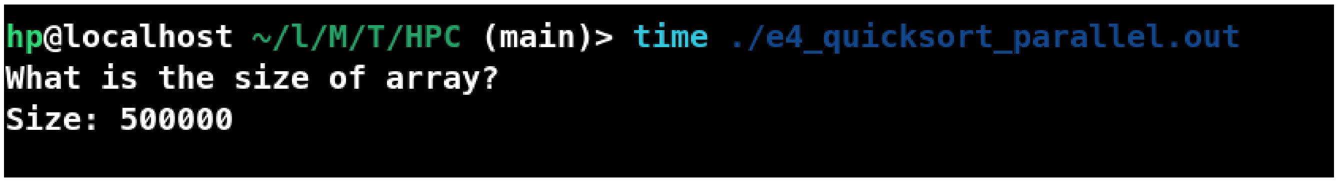
500k records:



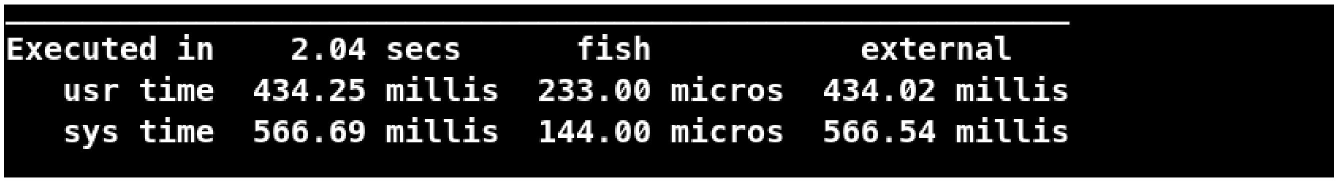
```
hp@localhost ~/l/M/T/HPC (main)> time ./e4_quicksort_serial.out
What is the size of array?
Size: 500000
```



```
________________________________________________________
Executed in    2.30 secs      fish           external
   usr time   95.49 millis  251.00 micros   95.24 millis
   sys time    6.50 millis  153.00 micros    6.35 millis
```

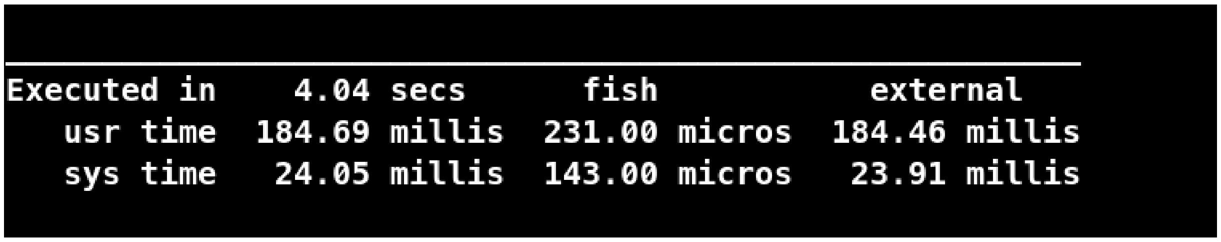


```
hp@localhost ~/l/M/T/HPC (main)> time ./e4_quicksort_parallel.out
What is the size of array?
Size: 500000
```



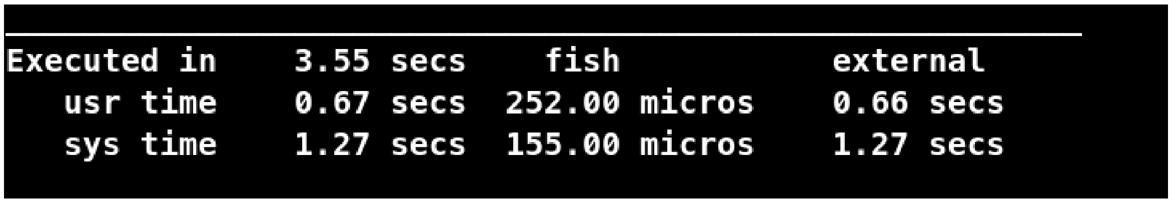
```
________________________________________________________
Executed in    2.04 secs      fish           external
   usr time  434.25 millis  233.00 micros  434.02 millis
   sys time  566.69 millis  144.00 micros  566.54 millis
```

1 Million Records:
Serial:



```
________________________________________________________
Executed in    4.04 secs      fish           external
   usr time  184.69 millis  231.00 micros  184.46 millis
   sys time   24.05 millis  143.00 micros   23.91 millis
```

Parallel:



```
________________________________________________________
Executed in    3.55 secs    fish           external
   usr time    0.67 secs  252.00 micros    0.66 secs
   sys time    1.27 secs  155.00 micros    1.27 secs
```