

[CSS553]

Name: Rahul Ranjan

Roll No: 20CS8016

Assignment 4

1. Observe and compare the physical time taken to (i) create 100,000 child processes (ii) create 20,000 child threads. Write two separate codes to perform the experiment. Use some system call to read the system clock time just before the creation starts and just after all the creation is over, display the difference in physical time.
Check if there is any significant speedup achieved in using threads which are light-weight processes.
2. Write a suitable code using `clone()` system call to speedup finding all prime numbers in a given range $[1, N]$, creating M child threads (LWPs) by the parent process thread.
First try to run it for two threads then generalize it for M child threads
3. Solve the Question 2 by using `pthread` library functions instead of `clone()` system call
4. Write a program using `pthread` library functions in which a parent process thread creates n child threads, such that even numbered child threads are created in attachable mode and odd numbered child threads are created in detachable mode. Show that the child threads created in attachable mode communicates by passing any value (integer) back to the parent using `pthread_exit()` function call.
5. Write a program in which a parent process creates a shared memory and puts an integer variable with initial value 1. Next, the parent creates ' n ' number of children and attach each children with the shared memory. The objective of each child will be to read the integer variable in the shared memory and increment it by one ' m ' number of times separately. The parent process will synchronize to the termination of all child processes and finally reads the final value of the shared integer variable and displays the value.
Run the program 10 times and observe if the value is equal to consistent each time it got executed?
6. Write a program in which a parent process will create a shared memory in which an array of 10 integers will be there (the initial value of the array elements will be 0). The parent will create a child and attach with the shared memory. The parent will fill up the odd indexed locations of the array by generating the random number (in between 1 to 100). The child will check the odd indexed locations and if it finds it as a non-zero integer then it increments the integer by 2 and store in the next adjacent even indexed location. Finally, the parent process will print the summation and average of integers stored in the even indexed locations (i.e., the integer that are included by the child process) and the child process will do the same for the integers stored in the odd indexed locations (i.e., the integer that are included by the parent process).

Answer 1 – Threads vs Process

→ 20000 Threads

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <sys/types.h>
4  #include <sys/wait.h>
5  #include <stdlib.h>
6  #include <stdbool.h>
7  #include <pthread.h>
8  #include <time.h>
9  #define CTC 20000
10
11 pthread_t *c_thread;
12
13 void * thread_process(void * args){
14     pthread_exit(0);
15 }
16
17 int main(void){
18     int c_thread_count = CTC;
19     int i;
20
21     clock_t t_start, t_end;
22     t_start = clock();
23
24     c_thread = (pthread_t*)malloc(sizeof(pthread_t)*c_thread_count);
25     for (i = 0; i < c_thread_count; i++){
26         pthread_create(&c_thread[i], NULL, thread_process, NULL);
27         pthread_detach(c_thread[i]);
28     }
29
30     t_end = clock();
31     double t_time_used = ((double)(t_end - t_start))/CLOCKS_PER_SEC;
32     printf("\nTime taken for m    threads: %lf\n\n", c_thread_count, t_time_used);
33     return 0;
34 }
```

>> Output – Threads Run 1

```
● gofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q1a_thread.c -o q1a_thread.out && ./q1a_thread.out
```

Time taken for 20000 threads: 0.586299

>> Output – Threads Run 2

```
● gofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q1a_thread.c -o q1a_thread.out && ./q1a_thread.out
```

Time taken for 20000 threads: 0.840054

➔ 100000 Processes

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <sys/types.h>
4  #include <sys/wait.h>
5  #include <stdlib.h>
6  #include <stdbool.h>
7  #include <pthread.h>
8  #include <time.h>
9  #define CPC 100000
10
11 pid_t *c_pid;
12
13 int main(void){
14     int c_process_count = CPC;
15     int i;
16
17     pid_t wait_p; int status;
18     clock_t p_start, p_end;
19     p_start = clock();
20
21     c_pid = (pid_t*)malloc(sizeof(pid_t)*c_process_count);
22     for (i = 0; i < c_process_count; i++){
23         c_pid[i] = fork();
24         if (c_pid[i] == 0){
25             exit(EXIT_SUCCESS);
26         }
27     }
28
29     // while((wait_p = wait(&status)) > 0);
30     p_end = clock();
31     double p_time_used = ((double)(p_end - p_start))/CLOCKS_PER_SEC;
32     printf("\nTime taken for m    process: %lf\n\n", c_process_count, p_time_used);
33
34     return 0;
35 }
```

>> Output – Processes Run 1

```
● gofynugt@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q1b_process.c -o q1b_process.out && ./q1b_process.out
```

Time taken for 100000 process: 0.581557

>> Output – Processes Run 2

```
● gofynugt@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q1b_process.c -o q1b_process.out && ./q1b_process.out
```

Time taken for 100000 process: 0.655198

Answer 2 – Find primes using clone system call (Multiprocessing)

```
1  #define _GNU_SOURCE
2  #include <sched.h>
3  #include <sys/syscall.h>
4  #include <linux/sched.h>
5  #include <stdio.h>
6  #include <unistd.h>
7  #include <sys/types.h>
8  #include <sys/wait.h>
9  #include <stdlib.h>
10 #include <stdbool.h>
11 #include <string.h>
12 #define STACK_SIZE 32784
13 #define CC 5
14
15 typedef struct c_args {
16     int x, y, t_i;
17     void *stack;
18 } c_args;
19
20 bool isPrime(int n){
21     if (n <= 1)
22         return false;
23     for (int i = 2; i*i <= n; i++)
24         if (n % i == 0) return false;
25     return true;
26 }
27
28 int getPrimes(void * a){
29     c_args * args = (c_args*)a;
30     int start=(int)((c_args*)args)->x;
31     int end=(int)((c_args*)args)->y;
32
33     for (int i = start; i <= end; i++){
34         if(isPrime(i)){
35             FILE *log = fopen("primes.txt", "a+");
36             fprintf(log, "%d ", i);
37             fclose(log);
38         }
39     }
40
41     return 0;
42 }
43
```

```

44 int main(void){
45
46     printf("\nEnter #x and #y: ");
47     int x,y;
48     scanf("%d", &x);
49     scanf("%d", &y);
50     printf("\n");
51     int start, end, cc = CC;
52     // Resetting the logfile
53     FILE * log;
54     log = fopen("primes.txt", "w");
55     fclose(log);
56
57     c_args **args = (c_args**) malloc(sizeof(c_args*));
58     for (int i = 0; i < CC; i++){
59         args[i] = (c_args*) malloc(sizeof(c_args));
60         args[i]->x = i*(y-x)/cc + x+1;
61         args[i]->y = (i+1)*(y-x)/cc + x;
62         args[i]->stack = malloc(STACK_SIZE);
63         args[i]->t_i = i;
64         clone(&getPrimes, (args[i]->stack)+STACK_SIZE, CLONE_VM, args[i]);
65     }
66
67     return 0;
68 }

```

>> Output



primes.txt

```

31 191 347 109 269 37 271 349 113 193 277 41 353 127 197
43 359 281 131 199 283 47 367 137 211 293 53 373 223 139
307 59 379 227 149 311 61 383 229 151 313 67 389 233 157
317 397 71 239 163 401 331 241 73 167 409 251 79 337 173
83 419 257 179 89 263 181 97 101 103 107

```

Answer 3 – Find primes using multithreads

```
1  #include <linux/sched.h>
2  #include <sys/syscall.h>
3  #include <stdio.h>
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7  #include <stdlib.h>
8  #include <stdbool.h>
9  #include <pthread.h>
10 #include <string.h>
11 #define TC 5
12
13 pthread_t c_thread[TC];
14
15 typedef struct c_args {
16     int x, y, t_i;
17 } c_args;
18
19 bool isPrime(int n){
20     if (n <= 1)
21         return false;
22     for (int i = 2; i*i <= n; i++)
23         if (n % i == 0) return false;
24     return true;
25 }
26
27 void * getPrimes(void * a){
28     c_args * args = (c_args*)a;
29     int start = (int)((c_args*)args)->x;
30     int end = (int)((c_args*)args)->y;
31     printf(">> @index: %d >> ", (int)((c_args*)args)->t_i);
32     for (int i = start; i <= end; i++)
33         if(isPrime(i))
34             printf("%d ", i);
35
36     printf("\n");
37     return 0;
38 }
39
40 int main(void){
41
42     printf("\nEnter #x and #y: ");
43     int x,y;
44     scanf("%d", &x);
45     scanf("%d", &y);
46     printf("\n");
47     int start, end, cc = TC;
48
49     c_args **args = (c_args**) malloc(sizeof(c_args)*TC);
50     for (int i = 0; i < TC; i++){
51         args[i] = (c_args*) malloc(sizeof(c_args));
52         args[i]->x = i*(y-x)/cc + x+1;
53         args[i]->y = (i+1)*(y-x)/cc + x;
54         args[i]->t_i = i;
55         pthread_create(&c_thread[i], NULL, getPrimes, (void*)(args[i]));
56     }
57     int c;
58     for (int i = 0; i < TC; i++)
59         c = pthread_join(c_thread[i], NULL);
60     printf("\n");
61
62     return 0;
63 }
```

>> Output – Case 1 (n = 250)

- `goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q3_pthread.c -o q3_pthread.out && ./q3_pthread.out`

Enter #x and #y: 1 250

```
>> @index: 0 >> 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
>> @index: 4 >> 211 223 227 229 233 239 241
>> @index: 3 >> 151 157 163 167 173 179 181 191 193 197 199
>> @index: 1 >> 53 59 61 67 71 73 79 83 89 97
>> @index: 2 >> 101 103 107 109 113 127 131 137 139 149
```

>> Output – Case 2 (n = 300)

- `goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q3_pthread.c -o q3_pthread.out && ./q3_pthread.out`

Enter #x and #y: 1 300

```
>> @index: 3 >> 181 191 193 197 199 211 223 227 229 233 239
>> @index: 2 >> 127 131 137 139 149 151 157 163 167 173 179
>> @index: 1 >> 61 67 71 73 79 83 89 97 101 103 107 109 113
>> @index: 0 >> 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
>> @index: 4 >> 241 251 257 263 269 271 277 281 283 293
```

Answer 4 – Thread communication

```
1  #include <linux/sched.h>
2  #include <sys/syscall.h>
3  #include <stdio.h>
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7  #include <stdlib.h>
8  #include <stdbool.h>
9  #include <pthread.h>
10 #include <string.h>
11 #define TC 5
12
13 pthread_t c_thread[TC];
14
15 typedef struct c_args {
16     int x, y, t_i;
17 } c_args;
18
19 void *thread_function(void *args){
20     pthread_exit(0);
21 }
22
23 bool isEven(int i){
24     return (i%2 == 0)? 1: 0;
25 }
26
27 int main(void){
28
29     printf("\nEnter #n: ");
30     int n = TC;
31     scanf("%d", &n);
32     printf("\n");
33     int cc = n;
34
35     c_args **args = (c_args**) malloc(sizeof(c_args)*TC);
36     for (int i = 0; i < cc; i++){
37         args[i] = (c_args*) malloc(sizeof(c_args));
38         args[i]->t_i = i;
39         pthread_create(&c_thread[i], NULL, thread_function, (void*)(args[i]));
40         if (!isEven(i)){
41             pthread_detach(c_thread[i]);
42         }
43     }
44     int *c = (int*)malloc(sizeof(int)*n);
45     for (int i = 0; i < n; i++) c[i] = -1;
46     for (int i = 0; i < n; i++)
47         printf("%d", c[i]);
48     printf("\n");
49
50     for (int i = 0; i < n; i++)
51         if (isEven(i))
52             c[i] = pthread_join(c_thread[i], NULL);
53
54     for (int i = 0; i < n; i++)
55         printf("%d", c[i]);
56     printf("\n");
57
58     return 0;
59 }
```



>> Output

- `goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q4_detachAttach.c -o q4_detachAttach.out && ./q4_detachAttach.out`

Enter #n: 8

```
-1 -1 -1 -1 -1 -1 -1 -1  
0 -1 0 -1 0 -1 0 -1
```

 // Array of values initiated with -1

 // Returning status 0 from attached threads

Answer 5 – Shared variable between processes.

```
1  #include <linux/sched.h>
2  #include <sys/syscall.h>
3  #include <stdio.h>
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7  #include <stdlib.h>
8  #include <stdbool.h>
9  #include <pthread.h>
10 #include <string.h>
11 #include <sys/ipc.h>
12 #include <sys/shm.h>
13 #include <errno.h>
14 #define SHM_KEY 0x1234
15 #define TC      5
16
17 pid_t *c_pid, wait_p;
18
19 int main(void){
20
21     int shmid = shmget(SHM_KEY, sizeof(int), 0666|IPC_CREAT);
22     if (shmid == -1){
23         perror("Shared Memory");
24         return 1;
25     }
26
27     int n, m;
28     scanf("%d %d", &n, &m);
29
30     void *memory = shmat(shmid, NULL, 0);
31     int *X = (int*)memory;
32     *X = 1;
33     int status;
34     c_pid = (pid_t*) malloc(sizeof(pid_t)*n);
35     for (int i = 0; i < n; i++){
36         c_pid[i] = fork();
37         if (c_pid[i] == 0){
38             void *mem_c = shmat(shmid, NULL, 0);
39             int *C = (int*)mem_c;
40             (*C) += m;
41             printf(">> Updated to: M @PID: ] \n", (*C), getpid());
42             exit(EXIT_SUCCESS);
43         }
44     }
45
46     while((wait_p = wait(&status)) > 0);
47     printf("\nFinal Value: %d\n\n", *X);
48
49     return 0;
50 }
```

>> Output – Run 1

```
● goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/
mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q5_sharedMem.
c -o q5_sharedMem.out && ./q5_sharedMem.out
20 200
>> Updated to: 201 @PID: 28533
>> Updated to: 401 @PID: 28534
>> Updated to: 601 @PID: 28535
>> Updated to: 801 @PID: 28540
>> Updated to: 1001 @PID: 28538
>> Updated to: 1201 @PID: 28537
>> Updated to: 1401 @PID: 28541
>> Updated to: 1601 @PID: 28536
>> Updated to: 1801 @PID: 28546
>> Updated to: 2001 @PID: 28549
>> Updated to: 2201 @PID: 28545
>> Updated to: 2401 @PID: 28547
>> Updated to: 2601 @PID: 28539
>> Updated to: 2801 @PID: 28551
>> Updated to: 3201 @PID: 28544
>> Updated to: 3401 @PID: 28552
>> Updated to: 3601 @PID: 28550
>> Updated to: 3001 @PID: 28548
>> Updated to: 3801 @PID: 28543
>> Updated to: 4001 @PID: 28542

Final Value: 4001
```

>> Output – Run 2

```
● goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/
mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q5_sharedMem.
c -o q5_sharedMem.out && ./q5_sharedMem.out
20 200
>> Updated to: 201 @PID: 28615
>> Updated to: 401 @PID: 28616
>> Updated to: 601 @PID: 28618
>> Updated to: 801 @PID: 28617
>> Updated to: 1001 @PID: 28619
>> Updated to: 1201 @PID: 28624
>> Updated to: 1401 @PID: 28623
>> Updated to: 1601 @PID: 28620
>> Updated to: 1801 @PID: 28622
>> Updated to: 2201 @PID: 28621
>> Updated to: 2401 @PID: 28625
>> Updated to: 2601 @PID: 28630
>> Updated to: 2001 @PID: 28626
>> Updated to: 2801 @PID: 28633
>> Updated to: 3001 @PID: 28629
>> Updated to: 3201 @PID: 28634
>> Updated to: 3401 @PID: 28628
>> Updated to: 3601 @PID: 28632
>> Updated to: 3801 @PID: 28627
>> Updated to: 4001 @PID: 28631

Final Value: 4001
```

>> Output – Run 3

```
● goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/
mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q5_sharedMem.
c -o q5_sharedMem.out && ./q5_sharedMem.out
20 200
>> Updated to: 201 @PID: 28697
>> Updated to: 401 @PID: 28698
>> Updated to: 601 @PID: 28699
>> Updated to: 801 @PID: 28702
>> Updated to: 1001 @PID: 28701
>> Updated to: 1201 @PID: 28700
>> Updated to: 1401 @PID: 28710
>> Updated to: 1801 @PID: 28711
>> Updated to: 1801 @PID: 28709
>> Updated to: 2001 @PID: 28708
>> Updated to: 2401 @PID: 28713
>> Updated to: 2601 @PID: 28703
>> Updated to: 2801 @PID: 28716
>> Updated to: 3001 @PID: 28707
>> Updated to: 2201 @PID: 28704
>> Updated to: 3201 @PID: 28712
>> Updated to: 3601 @PID: 28706
>> Updated to: 3801 @PID: 28714
>> Updated to: 3401 @PID: 28715
>> Updated to: 4001 @PID: 28705

Final Value: 4001
```

>> Output – Run 4

```
● goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs$ cd "/
mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q5_sharedMem.
c -o q5_sharedMem.out && ./q5_sharedMem.out
20 200
>> Updated to: 201 @PID: 28779
>> Updated to: 401 @PID: 28780
>> Updated to: 601 @PID: 28781
>> Updated to: 801 @PID: 28783
>> Updated to: 1201 @PID: 28784
>> Updated to: 1001 @PID: 28782
>> Updated to: 1401 @PID: 28785
>> Updated to: 1601 @PID: 28790
>> Updated to: 1801 @PID: 28792
>> Updated to: 2201 @PID: 28791
>> Updated to: 2401 @PID: 28789
>> Updated to: 2801 @PID: 28788
>> Updated to: 3001 @PID: 28798
>> Updated to: 2601 @PID: 28793
>> Updated to: 3401 @PID: 28797
>> Updated to: 3601 @PID: 28787
>> Updated to: 3801 @PID: 28786
>> Updated to: 4001 @PID: 28796
>> Updated to: 3201 @PID: 28795
>> Updated to: 2001 @PID: 28794

Final Value: 4001
```

It remains consistent for the rest of the runs too. No race condition occurring.

Answer 6 – Shared Array between Parent and Child

```
1  #include <linux/sched.h>
2  #include <sys/syscall.h>
3  #include <stdio.h>
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7  #include <stdlib.h>
8  #include <stdbool.h>
9  #include <pthread.h>
10 #include <string.h>
11 #include <sys/ipc.h>
12 #include <sys/shm.h>
13 #include <errno.h>
14 #define SHM_KEY 0x1236
15 #define TC      5
16 #define SIZE    10
17
18 pid_t c_pid, wait_p;
19
20 bool isOdd(int i){
21     return (i%2 != 0)? 1: 0;
22 }
23
24 int main(void){
25     int n = SIZE;
26     int shmid = shmget(SHM_KEY, n*sizeof(int), 0666 | IPC_CREAT);
27     if (shmid == -1){
28         perror("Shared Memory");
29         return 1;
30     }
31     void *memory = shmat(shmid, NULL, 0);
32     int *X = (int*)memory;
33     for (int i = 0; i < n; i++) X[i] = 0;
34     int status;
35     double sum, average;
36     for (int i = 0; i < n; i++)
37         if (isOdd(i))
38             X[i] = rand()%100 + 1;
39     printf("\nRandomly Genarated Array: ");
40     for (int i = 0; i < n; i++)
41         printf("%d ", X[i]);
42     printf("\n");
43
44     c_pid = fork();
45     if (c_pid == 0){
46         void *c_mem = shmat(shmid, NULL, 0);
47         int *C = (int*)c_mem;
```

```

48     for (int i = 0; i+1 < n; i++)
49         if (isOdd(i))
50             if (C[i] != 0)
51                 C[i+1] = C[i]+2;
52     for (int i = 0; i < n; i++)
53         if (isOdd(i))
54             sum += C[i];
55     average = sum/n;
56     printf("Array in child process:  ");
57     for (int i = 0; i < n; i++)
58         printf("%d ", C[i]);
59     printf("\n");
60     printf("\nChild >> Sum: %.2lf\nChild >> Average: %.2lf\n", sum, average);
61     exit(EXIT_SUCCESS);
62 }
63 while((wait_p = wait(&status)) > 0);
64 for (int i = 0; i < n; i++)
65     if (!isOdd(i))
66         sum += X[i];
67     average = sum/n;
68     printf("\nParent >> Sum: %.2lf\nParent >> Average: %.2lf\n", sum, average);
69     return 0;
70 }

```

>> Output

- goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System Labs\$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs/Assignment 4" && gcc q6_oddEven.c -o q6_oddEven.out && ./q6_oddEven.out

```

Randomly Genarated Array:  0 84  0 87  0 78  0 16  0 94
Array in child process:    0 84 86 87 89 78 80 16 18 94

```

```

Child >> Sum: 359.00
Child >> Average: 35.90

```

```

Parent >> Sum: 273.00
Parent >> Average: 27.30

```

All compiler codes are uploaded [here](#).