lab04_ex1.c

```c
/** Daniel Hanna
   CSC 345-01
   Lab 04 Exercise 1 */

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

int main(int argc, char** argv) {
    int n = atoi(argv[1]);              /* get argument input */
    int i, j;
    int count  = 0;
    time_t begin = time(NULL);        /* keep track of time */
    pid_t id = getpid();              /* get pid value */

    for(i = 1; i <= n; ++i) {         /* find all primes */
        for(j = 2; j < i; ++j) {
            if (i % j == 0) {
                break;
            }
        }
        if (j == i) {              /* increment count when i and j match */
            ++count;
        }
    }
    printf("\n");
    printf("* Process %d found %d primes within [1, %d] in %ld
seconds\n", id, count, n, time(NULL)-begin);
    return 0;

}
```

lab04_ex1.c Results

```
osc@osc-VirtualBox: ~/work/lab04

File  Edit  View  Search  Terminal  Help
osc@osc-VirtualBox:~/work/lab04$ ./lab04_ex1 30000 50

* Process 24660 found 3245 primes within [1, 30000] in 0 seconds
osc@osc-VirtualBox:~/work/lab04$ ./lab04_ex1 300000 &
[1] 24666
osc@osc-VirtualBox:~/work/lab04$ ./lab04_ex1 300000

* Process 24666 found 25997 primes within [1, 300000] in 25 seconds

* Process 24667 found 25997 primes within [1, 300000] in 26 seconds
[1]+  Done                    ./lab04_ex1 300000
osc@osc-VirtualBox:~/work/lab04$
```

lab04_ex2.c

```c
/** Daniel Hanna
    CSC 345-01
    Lab 04 Exercise 2 */


#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS 5


void *runner(void *param) {
    pthread_exit(0);
}


int main(int argc, char *argv[]) {
    int i, policy;
    pthread_t tid[NUM_THREADS];
    pthread_attr_t attr;


    pthread_attr_init(&attr);
```

```c
/* get the scheduling policy */
if(pthread_attr_getschedpolicy(&attr, &policy) != 0) {
    fprintf(stderr, "Unable to get policy.\n");
}

/* output what the policy is */
else {
    if(policy == SCHED_OTHER) {
        printf("SCHED_OTHER\n");
    }
    else if(policy == SCHED_RR) {
        printf("SCHED_RR\n");
    }
    else if(policy == SCHED_FIFO) {
        printf("SCHED_FIFO\n");
    }
}

/* try setting the scheduling policy */
if(pthread_attr_setschedpolicy(&attr, SCHED_FIFO) != 0) {
    fprintf(stderr, "Unable to set policy.\n");
}

/* after setting policy, get it & see what it is */
if(pthread_attr_getschedpolicy(&attr, &policy) != 0) {
    fprintf(stderr, "Unable to get policy.\n");
}

else {
    if(policy == SCHED_OTHER) {
        printf("SCHED_OTHER\n");
    }
    else if(policy == SCHED_RR) {
        printf("SCHED_RR\n");
    }
```
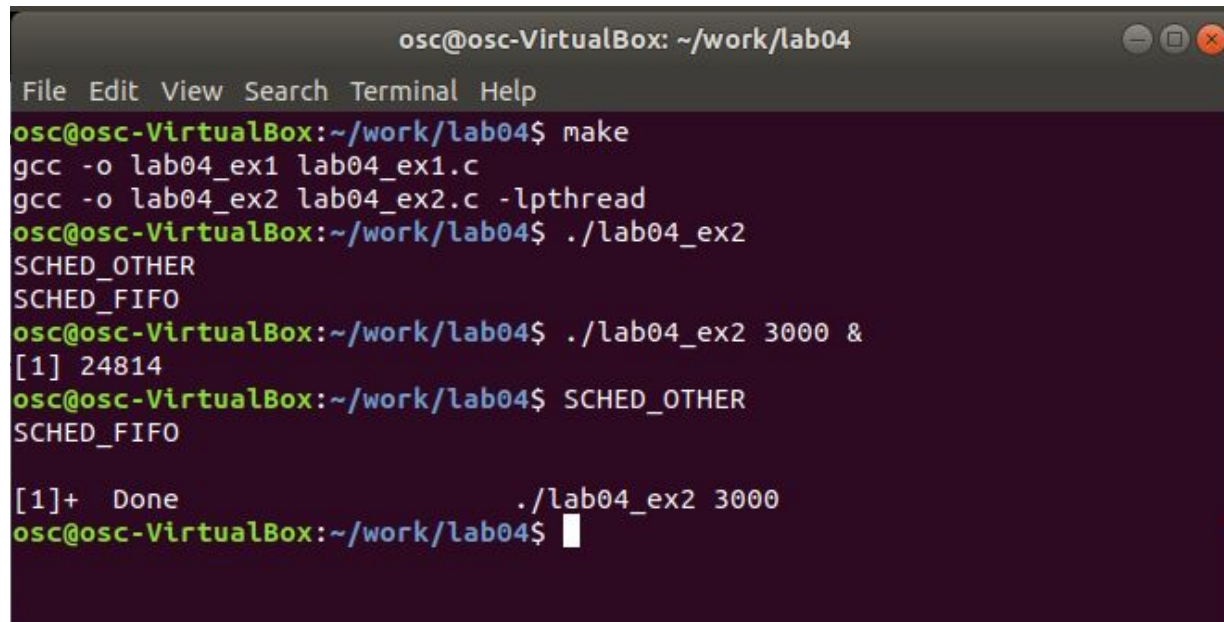
```
        else if(policy == SCHED_FIFO) {
            printf("SCHED_FIFO\n");
        }
    }
    for(i = 0; i<NUM_THREADS; i++) {
        pthread_create(&tid[i], &attr, runner, NULL);
    }
    for(i = 0; i<NUM_THREADS; i++) {
        pthread_join(tid[i], NULL);
    }
}
```

lab04_ex2.c Results