CS 4323 Design and Implementation of Operating Systems I

Assignment 02: Full Marks 100

Due Date: 02/25/2024, 11:59 PM CT **End Date: 03/03/2024, 11:59 PM CT**

- This assignment needs to be done individually and it should be computer typed, not handwritten.
- Any formula used on the numerical question(s) must be clearly stated. Also, appropriate units must be clearly specified whenever applicable. Failure to mention these will deduct up to 50% points.
- The programs need to be tested on CSX servers. Instructions on how to access CSX server is available in
  https://cas.okstate.edu/department_of_computer_science/it_services/logging_on.html
- Assignment needs to be submitted on Canvas, not on CSX server. Note the difference between submission and testing, mentioned in the previous point.
- No email submissions will be accepted.

Note: To run the codes, you need to add appropriate header(s) and/or complete the missing part.

1. Based on the following snippet of C code and considering none of the *fork()* and *pthread()* calls fail, answer the following:
   a) Draw the process tree diagram showing all the threads (with process ID and thread ID) created in the program. Before each thread and process terminates, what are the final values for the following variables: *global_var*, *static_var*, *local_var*, *dynamic_var*. **[8 Marks]**
   b) Verify your answer in part(a) by adding appropriate *printf* statement(s) in the above code and explain the output. When you submit the c file for this question, please name it as Question1.c. **[2 + 2 Marks]**

```c
int global_var = 0;

static int static_var = 0;

void* thread_function(void* arg) {
    int local_var = 0;

    int* dynamic_var = (int*)malloc(sizeof(int));
    *dynamic_var = 0;

    local_var++;
    (*dynamic_var)++;
    global_var++;
    static_var++;

    free(dynamic_var);

    pthread_exit(NULL);
}

int main() {
    int i;
    pid_t pid;

    for (i = 0; i < 3; i++) {
        pid = fork();
        if (pid == 0) {
            pthread_t threads[3];
            int j;
            for (j = 0; j < 3; j++) {
                pthread_create(&threads[j], NULL, thread_function,
NULL);
            }
            for (j = 0; j < 3; j++) {
                pthread_join(threads[j], NULL);
            }
            exit(0);
        } else if (pid < 0) {
            fprintf(stderr, "Fork failed\n");
            return 1;
        }
    }

    for (i = 0; i < 3; i++) {
        wait(NULL);
    }

    return 0;
}
```

2. Based on the following snippet of C code and considering none of the *fork()* and *pthread()* calls fail, answer the following:
   a) Draw the process tree diagram showing all the threads (with process ID and thread ID) created in the program. Before each thread and process terminates, what are the final values for the following variables: *global_var*, *static_var*, *local_var*, *dynamic_var*. **[8 Marks]**
   b) Verify your answer in part(a) by adding appropriate *printf* statement(s) in the given code and explain the output. When you submit the c file for this question, please name it as Question2b.c. **[2 + 2 Marks]**
   c) How will you modify the given code, if you would like each thread to have its own instance of *static_var* variable? Explain your reasoning and submit the appropriate C file. When you submit the c file for this question, please name it as Question2c.c. **[2 + 3 Marks]**

```c
int global_var = 0;

static int static_var = 0;

void* thread_function(void* arg) {
    int local_var = 0;

    int* dynamic_var = (int*)malloc(sizeof(int));
    *dynamic_var = 0;

    local_var++;
    (*dynamic_var)++;
    global_var++;
    static_var++;

    free(dynamic_var);

    pid_t pid;
    int i;
    for (i = 0; i < 3; i++) {
        pid = fork();
        if (pid == 0) {
            local_var++;
            (*dynamic_var)++;
            global_var++;
            static_var++;

            exit(0);
        } else if (pid < 0) {
            fprintf(stderr, "Fork failed\n");
            return NULL;
        }
    }

    for (i = 0; i < 3; i++) {
        wait(NULL);
    }

    pthread_exit(NULL);
}

int main() {
    pthread_t threads[3];
    int i;

    for (i = 0; i < 3; i++) {
        pthread_create(&threads[i], NULL, thread_function, NULL);
    }

    for (i = 0; i < 3; i++) {
        pthread_join(threads[i], NULL);
    }

    return 0;
}
```

3. Consider a code snippet on C as shown below. Assuming none of the *fork()* and the *pthread_create()* system call fail, answer the following:
   a) Specify all the possible outputs from the program and explain them. **[6 Marks]**
   b) List all the issues with the given code and specify the solution to fix it. Provide the C file that includes the solutions and explain the solution. Name the C file as Question3.c. **[10 Marks]**

```c
void *thread_function1(void *arg) {
    printf("Thread 1 created\n");

    /* Thread does something meaningful task..*/

    char *argv[] = {"ls", "-l", NULL};
    execvp("ls", argv);

    perror("execvp");
    printf("Thread 1 is done\n");
}

void *thread_function2(void *arg) {
    printf("Thread 2 created\n");

    /* Thread does something meaningful task..*/

    printf("Thread 2 is done\n");
    pthread_exit(NULL);
}

int main() {
    pid_t pid;
    pthread_t thread_id;

    printf("Main process started\n");

    if (pthread_create(&thread_id, NULL, thread_function1, NULL) != 0){
        perror("pthread_create");
        exit(EXIT_FAILURE);
    }

    if (pthread_create(&thread_id, NULL, thread_function2, NULL) != 0){
        perror("pthread_create");
        exit(EXIT_FAILURE);
    }

    pid = fork();
    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        printf("Child process executing\n");
    } else {
        printf("Parent process continuing\n");
    }

    if (pthread_join(thread_id, NULL) != 0) {
        perror("pthread_join");
        exit(EXIT_FAILURE);
    }

    printf("Main process is done\n");

    return 0;
}
```

4. Complete the tasks specified inside the block comment in the given C code below. Explain each of the program output. Name the C file as Question4.c. **[10 + 5 Marks]**

```c
#define NUM_THREADS 5

int global_var = 0;

static int static_var = 0;

void *thread_func(void *arg) {

    for (int i = 0; i < 100000; ++i) {
        // Increment local variable
        local_var++;    // this is passed from the main function

        // Increment global variable
        global_var++;

        // Increment static variable
        static_var++;

        // Increment dynamic variable
        (*dynamic_var)++;
    }

    printf("Thread finished: Local var: %d, Global var: %d, Static 
var: %d, Dynamic var: %d\n", local_var, global_var, static_var, 
*dynamic_var);

    /* free the dynamic variable */
    /* exit the thread */
}

int main() {
    pthread_t threads[NUM_THREADS];

    int local_var = 42;

    for (int i = 0; i < NUM_THREADS; ++i) {

  /* Create a thread and pass the variable "local_var" to it. */
  /* Make sure to check if the thread creation is successful or not. */

    }

 /* Join all the threads */
 /* Make sure to check if the thread are joined successfully or not. */

    printf("Main thread finished. Local var: %d, Global var: %d, Static 
var: %d\n", local_var, global_var, static_var);

    return 0;
}
```

5. What is the output in the following C code? Explain your reason. **[10 Marks]**

```
int value = 0;

void *func1(void *param) {
    value++;
    printf("alue = %d\n" , value);
    pthread_exit(0);
}

int main(int argc, char *argv[]){
    int pid;
    pthread_t tid1, tid2;

    pthread_create(&tid1,NULL,func1,NULL);
    pthread_create(&tid2,NULL,func1,NULL);

    pthread_join(tid1,NULL);
    pthread_join(tid2,NULL);

    return 0;
}
```

6. What is the output in the following C code? Explain your reason. **[10 Marks]**

```
int count = 10;

void fun1_1(){
    count ++;
    printf("Thread1 (inside fun1_1: Count value is = %d\n", count);
}

void *fun1(void *arg){
    count ++;
    printf("Thread1 (before fun1_1 call): Count value is = %d\n",
count);
    fun1_1();
    printf("Thread1 (after fun1_1 call): Count value is = %d\n",
count);
    return NULL;
}

void *fun2(void *arg){
    count ++;
    printf("Thread2: Count value is = %d\n", count);
    return NULL;
}

int main(){
    pthread_t thread1, thread2;
    void *result;

    pthread_create(&thread1, NULL, fun1, NULL);
    pthread_create(&thread2, NULL, fun2, NULL);

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

    return 0;
}
```

7. Suppose a program consists of 50% sequential instructions and 50% parallelizable instructions. The sequential instructions take 40 ns to execute, and the parallelizable instructions take 60 ns to execute on a single processor. If the program runs on a system with 4 processors, what is the speedup gain achieved over running it on a single processor? **[10 Marks]**

8. Consider a computer system with a single processor. A portion of the program consists of sequential instructions that take 100 ns to execute, and the remaining portion consists of parallelizable instructions that take 200 ns to execute. The sequential portion accounts for 25% of the total execution time. If this program is run on a computer system with four processors, what is the speedup achieved. **[10 Marks]**

Submission Guidelines:

- Submit a single pdf file with the name:
    - assignment02_lastName_firstName.pdf

- You need to submit the C file for questions 1, 2, 3 and 4 and name them as specified in those questions.
    - Please make sure the submitted C files compile (without any warning) and run on the CSX server. If there is any error or warning during the compilation or at the runtime, 100% of the marks will be deducted for the given question.
    - Codes without comments will not receive any point.

- You need to include a single readMe.txt file which should include:
    - how to run your programs for question 1, 2, 3 and 4 with detail instruction,
    - on which server the program is tested.

- In each .c file, use the following header information:
    - Author Name:
    - Email: <Use only official email address>
    - Date:
    - Program Description:

- Handwritten assignment will receive 10 point penalty.