CS450/599: Parallel Programming

Assignment 1: Pthreads

Due date: see BBLearn

Preliminaries

You are expected to do your own work on all homework assignments. You may (and are encouraged to) engage in discussions with your classmates regarding the assignments, but specific details of a solution, including the solution itself, must always be your own work. See the academic dishonesty policy in the course syllabus.

On Written (non-programming) Assignments and Questions

These questions are used to test how well you understand the concepts we have discussed in class. Concurrent programs have non-deterministic execution and other irregularities that you will not see in sequential programs. It is important that you explain your answer with as much detail as needed. For instance, a result to a question may be straightforward, like: "the output of the program is 10". This is acceptable for such programs/questions. However, in other cases the output may not be as straightforward, as the question may be more opened ended; therefore, in such cases, you should provide a more detailed account of what you think the program is doing.

Submission Instructions

You should turn in a pdf called CS450_aX_NAME.pdf, where "NAME" is your NAU username and "X" is the assignment number (e.g., CS450_a1_mg1234.pdf). Assignments need to be turned in via BBLearn.

Grading

All questions are graded equally.

Question 1

Below is code that uses Pthreads. The program prints to the screen 4 lines. Assume these print statements are unbuffered. Answer the following regarding the program output:
True/False: The code below has one possible output. If false, list 3 possible program outputs.

```
void *do_work1(void *arg);
void *do_work2(void *arg);

int main(int argc, char *argv) {
  int array[100];
  int i,sum;
  pthread_t worker_thread1;
  pthread_t worker_thread2;
  struct arguments *arg;


  // Create thread1
```

```
  if (pthread_create(&worker_thread1, NULL,
                     do_work1, (void *)arg)) {
    fprintf(stderr,"Error while creating thread\n");
    exit(1);
  }

  // Create thread2
  if (pthread_create(&worker_thread2, NULL,
                     do_work2, (void *)arg)) {
    fprintf(stderr,"Error while creating thread\n");
    exit(1);
  }

  // Join with thread
  if (pthread_join(worker_thread1, NULL)) {
    fprintf(stderr,"Error while joining with child thread\n");
    exit(1);
  }

  if (pthread_join(worker_thread2, NULL)) {
    fprintf(stderr,"Error while joining with child thread\n");
    exit(1);
  }

  exit(0);
}

void *do_work1(void *arg) {
  printf("\nStarting1");
  printf("\nExiting1");

  return NULL;
}

void *do_work2(void *arg) {
  printf("\nStarting2");
  printf("\nExiting2");

  return NULL;
}
```

Question 2

Below is code that uses Pthreads. The program prints to the screen 2 lines. Assume these print statements are unbuffered. Answer the following regarding the program output:
True/False: The code below has one possible output. If false, what are the possible program outputs?

```
struct arguments {
  int value;
};
```

```
void *do_work(void *arg);

int main(int argc, char *argv) {
  pthread_t worker_thread1;
  pthread_t worker_thread2;
  struct arguments *arg[2];

  // Build argument to threads
  arg[0] = (struct arguments *)calloc(1, sizeof(struct arguments));
  arg[0]->value=5;

  arg[1] = (struct arguments *)calloc(1, sizeof(struct arguments));
  arg[1]->value=12;


  // Create thread1
  if (pthread_create(&worker_thread1, NULL,
                     do_work, (void *)arg[0])) {
    fprintf(stderr,"Error while creating thread\n");
    exit(1);
  }

  // Create thread2
  if (pthread_create(&worker_thread2, NULL,
                     do_work, (void *)arg[1])) {
    fprintf(stderr,"Error while creating thread\n");
    exit(1);
  }

  // Join with thread
  if (pthread_join(worker_thread1, NULL)) {
    fprintf(stderr,"Error while joining with child thread\n");
    exit(1);
  }

  if (pthread_join(worker_thread2, NULL)) {
    fprintf(stderr,"Error while joining with child thread\n");
    exit(1);
  }

  exit(0);
}

void *do_work(void *arg) {
  struct arguments *argument;
  argument=(struct arguments*)arg;

  int val=argument->value;
  printf("%d\n",val);
```

```
  return NULL;
}
```

Question 3

Below is code that uses Pthreads. The program prints to the screen 1 line. Assume these print statements are unbuffered. Answer the following regarding the program output:
True/False: The code below has one possible output. If false, what are the possible program outputs?

```
struct arguments {
  int *sum;
  int value;
};

void *do_work(void *arg);

int main(int argc, char *argv) {
  pthread_t worker_thread1;
  pthread_t worker_thread2;
  struct arguments *arg[2];

  int sum=0;

  // Build argument to threads
  arg[0] = (struct arguments *)calloc(1, sizeof(struct arguments));
  arg[0]->value=3;
  arg[0]->sum=&sum;

  arg[1] = (struct arguments *)calloc(1, sizeof(struct arguments));
  arg[1]->value=5;
  arg[1]->sum=&sum;

  // Create thread1
  if (pthread_create(&worker_thread1, NULL,
                     do_work, (void *)arg[0])) {
    fprintf(stderr,"Error while creating thread\n");
    exit(1);
  }

  // Create thread2
  if (pthread_create(&worker_thread2, NULL,
                     do_work, (void *)arg[1])) {
    fprintf(stderr,"Error while creating thread\n");
    exit(1);
  }

  // Join with thread
  if (pthread_join(worker_thread1, NULL)) {
    fprintf(stderr,"Error while joining with child thread\n");
    exit(1);
```

```
  }

  if (pthread_join(worker_thread2, NULL)) {
    fprintf(stderr,"Error while joining with child thread\n");
    exit(1);
  }

  printf("\nSum: %d",sum);

  exit(0);
}

void *do_work(void *arg) {
  struct arguments *argument;
  argument=(struct arguments*)arg;

  int val=argument->value;
  int *sum=argument->sum;

  *sum+=val;

  return NULL;
}
```

Question 4

Below is code that uses Pthreads. The program prints to the screen 2 lines. Assume these print statements are unbuffered. Answer the following regarding the program output:
True/False: The code below has one possible output. If false, what are the possible program outputs?

```
struct arguments {
  pthread_mutex_t *mutex;
  int val;
  int *count;
};

void *do_work(void *arg);

int main(int argc, char *argv) {
  pthread_t worker_thread1;
  pthread_t worker_thread2;
  pthread_mutex_t lock;
  pthread_mutex_init(&lock, NULL);

  struct arguments *arg[2];
  int count=0;


  // Build argument to threads
  arg[0] = (struct arguments *)calloc(1, sizeof(struct arguments));
  arg[0]->mutex=&lock;
```

```
  arg[0]->val=2;
  arg[0]->count=&count;

  arg[1] = (struct arguments *)calloc(1, sizeof(struct arguments));
  arg[1]->mutex=&lock;
  arg[1]->val=3;
  arg[1]->count=&count;


  // Create thread1
  if (pthread_create(&worker_thread1, NULL,
                     do_work, (void *)arg[0])) {
    fprintf(stderr,"Error while creating thread\n");
    exit(1);
  }

  // Create thread2
  if (pthread_create(&worker_thread2, NULL,
                     do_work, (void *)arg[1])) {
    fprintf(stderr,"Error while creating thread\n");
    exit(1);
  }

  // Join with thread
  if (pthread_join(worker_thread1, NULL)) {
    fprintf(stderr,"Error while joining with child thread\n");
    exit(1);
  }

  if (pthread_join(worker_thread2, NULL)) {
    fprintf(stderr,"Error while joining with child thread\n");
    exit(1);
  }

  exit(0);
}

void *do_work(void *arg) {
  struct arguments *argument;
  argument=(struct arguments*)arg;

  pthread_mutex_t *mutex=argument->mutex;
  int val=argument->val;
  int *count=argument->count;

  pthread_mutex_lock(mutex);
  *count+=val;
  printf("\n%d",*count);
  pthread_mutex_unlock(mutex);

  return NULL;
```

```
}
```

Question 5

Below is code that uses Pthreads. The program prints to the screen 2 lines. Assume these print statements are unbuffered. Answer the following regarding the program output:
True/False: The code below has one possible output. If false, what are the possible program outputs?

```
struct arguments {
  pthread_mutex_t *mutex;
  int val;
  int *count;
};

void *do_work(void *arg);

int main(int argc, char *argv) {
  pthread_t worker_thread1;
  pthread_t worker_thread2;
  pthread_mutex_t lock1;
  pthread_mutex_t lock2;
  pthread_mutex_init(&lock1, NULL);
  pthread_mutex_init(&lock2, NULL);

  struct arguments *arg[2];
  int count=0;


  // Build argument to threads
  arg[0] = (struct arguments *)calloc(1, sizeof(struct arguments));
  arg[0]->mutex=&lock1;
  arg[0]->val=2;
  arg[0]->count=&count;

  arg[1] = (struct arguments *)calloc(1, sizeof(struct arguments));
  arg[1]->mutex=&lock2;
  arg[1]->val=3;
  arg[1]->count=&count;




  // Create thread1
  if (pthread_create(&worker_thread1, NULL,
                     do_work, (void *)arg[0])) {
    fprintf(stderr,"Error while creating thread\n");
    exit(1);
  }
```

```
  // Create thread2
  if (pthread_create(&worker_thread2, NULL,
                     do_work, (void *)arg[1])) {
    fprintf(stderr,"Error while creating thread\n");
    exit(1);
  }

  // Join with thread
  if (pthread_join(worker_thread1, NULL)) {
    fprintf(stderr,"Error while joining with child thread\n");
    exit(1);
  }

  if (pthread_join(worker_thread2, NULL)) {
    fprintf(stderr,"Error while joining with child thread\n");
    exit(1);
  }


  exit(0);
}

void *do_work(void *arg) {
  struct arguments *argument;
  argument=(struct arguments*)arg;

  pthread_mutex_t *mutex=argument->mutex;
  int val=argument->val;
  int *count=argument->count;

  pthread_mutex_lock(mutex);
  *count+=val;
  printf("\n%d",*count);
  pthread_mutex_unlock(mutex);

  return NULL;
}
```